INTEGRATION, RENDERING AND EXTENSION OF TOOLS FOR THE SYNERGY BETWEEN NATURAL LANGUAGE TEXT AND PROCESS MODELS

Autor: Luis Delicado
Director: Josep Carmona
Codirector: Lluís Padró

Trabajo Final de Grado
Grado en Ingeniería en Informática
(Computación)

Facultat d'Informàtica de Barcelona

Junio 2016

Resumen

Con el incremento del desarrollo de herramientas para la descripción de procesos, Object Management Group (OMG) desarrollo Business Process Model and Notation para describir procesos de negocio y así poderlos implementar más fácilmente y estandarizar el formato de descripción de procesos de negocio. Antes de que apareciese esta notación, y también actualmente, la descripción de los procesos se hacían básicamente mediante textos, normalmente con una serie de instrucciones, los cuales no eran entendibles para los ordenadores.

Teniendo en mente estas dos formas de describir un proceso de negocio, nace este proyecto, en el que se ha creado una herramienta para poder convertir de uno a otro los dos tipos de descripción de una manera automática y validar la creación de éstos. En el proyecto se han desarrollado una serie de herramientas para trabajar con las descripciones de los procesos de negocios para posteriormente integrarlas en una interfaz web la cual facilite el uso de éstas y renderice los resultados de una manera fácil de entender.

Abstract

With the increasing development of tools for process description, Object Management Group (OMG) develops Business Process Model and Notation for describe business processes and implement them easily and standardizes the description format of business processes. Before this notation appeared, and also nowadays, the process description is done basically with texts, usually with instructions, which are not understandable for computers.

This project was born following the idea of two describing process formats, whose objective is develop a tool to convert between both formats automatically and validate user descriptions. In this project it has been developed some tools to work with this business processes descriptions, and integrate them in a web interface to help the user to use and render the results in a easily understandable way.

Índice

Índice de figuras

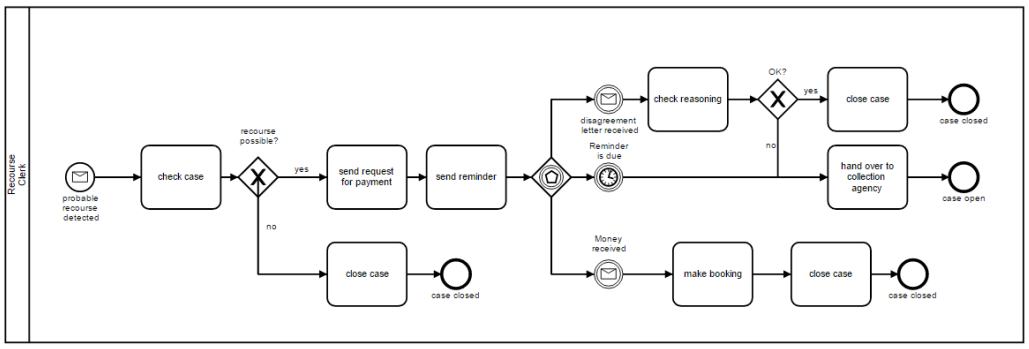
Índice de cuadros

1. Introducción

En los últimos años se han estado buscando y creando diferentes notaciones para la descripción de procesos de negocio. En 2005 se creó Business Process Model and Notation(BPMN) con el propósito de crear un estándar fácilmente legible y entendible por parte de la gente de negocios. Hoy en día la gran parte del público que trabaja con sistemas para descripción de modelos utiliza esta notación, por lo que ya existen muchos programas que ayudan a la creación de éstos, ya sea con un editor gráfico, ya que se pueden representar en forma de diagrama de flujo, u otros tipos de herramientas. Antes de esto, la forma usual de describir los procesos de negocio, la cual se sigue usando, era en un texto plano, normalmente una serie de instrucciones, el cual era entendible para personas pero para implementarlo no es sencillo.

El proyecto que se muestra a continuación, que se ha desarrollado como Trabajo Final de Grado, forma parte de un proyecto mayor desarrollado en combinación con otros TFG de otros compañeros. En la parte del proyecto desarrollada por otros compañeros se han llevado a cabo una serie de programas con los cuales facilitar la creación y la validación de dicho tipo de modelos de maneras que no existen en el mercado actual. Para ello, en los otros trabajos se han desarrollado tres módulos con dos objetivos diferentes.

Los dos primeros módulos tenían como objetivo la conversión de un modelo BPMN a un texto legible por cualquiera, como se puede ver en la figura 1, y al revés, pasándole un texto ser capaz de construir un modelo BPMN. Esto facilitaría la creación de modelos, al poder escribirlos en lenguaje natural, y la comunicación cuando se implementa, ya que se puede sacar un texto entendible a partir del modelo. El tercer módulo está más enfocado a la validación de modelos, ya que consiste en comparar un texto con un modelo BPMN para saber si ese modelo hace lo mismo o parecido a lo que se le pasa en el texto. De esta manera se puede comprobar que se ha creado un modelo con la idea que se tenía en mente.



For user recourse
clerk, starting with recourse
clerk gets the message sent by .After that, recourse
clerk check case. It ends with this choice will have 2 different endings:
- When recourse possible? with value yes, this sequence starts with recourse
clerk send request for payment. Subsequently, recourse
clerk send reminder. At the end, this choice will have 3 different endings:
* Ending 1, first, recourse
clerk receives a message and performs disagreement
letter received. Subsequently, recourse
clerk check reasoning. Finally different decision in each one of the 2 branches will give different endings:
- When ok? with value yes, recourse
clerk close case.
- The branch ok? under condition no, recourse
clerk hand over to collection agency.
* Ending 2, first, it's recourse
clerk. Finally recourse
clerk hand over to collection agency.
* The option 3 has as a consequence starting with recourse
clerk, after having received a message, does money
received. Subsequently, recourse
clerk make booking. It ends with recourse
clerk close case.
- The option recourse possible? as no has as a consequence recourse
clerk close case.

Figura 1: Ejemplo del conversor de BPMN a texto

La parte del proyecto que se realiza en este Trabajo Final de Grado tiene dos objetivos principales. El primer objetivo es completar este grupo de módulos con la creación de un programa el cual sea capaz de a partir de dos diagramas compararlos y deducir si representan procesos parecidos dando un resultado entendible por los usuarios a los que está destinado el programa. El segundo objetivo consiste en integrar todos los módulos creados hasta el momento incluyendo el que se creará en este trabajo en una interfaz web a la que puedan entrar los usuarios y ejecutar de manera intuitiva y fácil todos los módulos.

Este documento está estructurado en 5 partes. En primer lugar se realiza una pequeña introducción del proyecto general y de la parte que se va a realizar en el Trabajo Final de Grado explicando sus objetivos y el estado del arte. En segundo lugar se presenta una sección en la que se habla del alcance del proyecto, la planificación temporal y económica, y la sostenibilidad del proyecto. En tercer lugar se explicará el desarrollo del módulo del com-

parador de modelos empezando con una explicación sobre en qué consisten los modelos BPMN, qué elementos forman un modelo BPMN y en qué se utilizan, después una explicación del algoritmo que se utiliza y cómo se ha implementado, y para finalizar una serie de ejemplos y experimentos realizados. En cuarto lugar se explica cómo se ha realizado la interfaz web desde una explicación de cada módulo y cómo se ha integrado en la web, después se explica todas las tecnologías asociadas a los módulos que se han integrado seguido del funcionamiento de la web más allá de los módulos para acabar con el proceso de configuración del servidor en el cual se ejecuta la aplicación. Por último, se extraen las conclusiones que se han sacado del proyecto y se comenta el trabajo de futuro.

1.1. Estado del arte

Este proyecto, que pretende crear una interfaz para poder generar diagramas BPMN para procesos de empresas a partir de un texto y a la inversa, o comparar si su BPMN es parecido a la idea que tenían en su texto, es algo nuevo que no se ha tratado mucho y no existe nada parecido en el mercado actualmente. Dicho esto, se verán qué tipos de herramientas relacionadas con los BPMN existen en comparación a la interfaz que se ha desarrollado y algunos métodos que se utilizan para comparar modelos.

La mayor parte de los programas que existen relacionados con los BPMN son editores que te facilitan la creación de éstos. Entre éstos se pueden encontrar tanto programas descargables como web, por ejemplo, bpmn.io [1] o Apromore [2], y otros que a parte de editor tienen otras aplicaciones, es el caso de Signavio [3], una herramienta que a parte de editar fácilmente nuestros modelos nos permite ejecutarlos con algunas herramientas para verificar que nuestro diagrama sea correcto.

Para la parte del comparador, el problema que se quiere resolver es básicamente un caso especifico de graph-matching. Este problema consiste en, dado un grafo $G = (V, E)$, encontrar un conjunto independiente de aristas o también llamado matching. Para ello las aristas del conjunto no deben tener vértices en común [4]. En el caso del problema planteado en este proyecto, las aristas van de elementos del primer grafo a elementos del segundo grafo y se puede repetir vértices.

A parte, el problema que se plantea en este proyecto ya se ha intentado resolver con algunos métodos. El método general para comparar procesos consisten en dos fases. En primer lugar, identificar qué actividades del proce-

so corresponden a las del otro proceso. Si éstos han sido creados con diferente nivel de detalle o en otro orden no es una tarea trivial. Para esta parte existen diferentes algoritmos que se pueden usar. Una vez identificadas, se mira la similitud dependiendo de una serie de propiedades. Estas propiedades tienen en cuenta la distancia entre nodos y si son conexos entre ellos. Al mirar longitud de caminos, puede ser muy costoso de calcular sobretodo si hay bucles en el proceso [5].

Otro método que se utiliza para resolver este problema es comparar la representación abstracta de dos modelos basado en binary behavioral relations. Si los dos modelos abstractos son isomorfos entonces estos son equivalente, por otra parte, si no son isomorfos podemos utilizar algún algoritmo de corrección de errores del matching para ver las diferencias [6].

2. Alcance, planificación y sotenibilidad

2.1. Alcance

En este apartado comentaremos los objetivos que se quieren realizar y los requisitos para alcanzarlos, los problemas que nos podemos encontrar a lo largo de este proyecto y la metodología que se seguirá en el desarrollo del trabajo junto a las herramientas y métodos para hacer el seguimiento de éste.

2.1.1. Objetivos y requisitos del trabajo

Este trabajo tiene dos objetivos. En primer lugar unificar los módulos desarrollados por los otros compañeros en una interfaz gráfica de manera que sea fácilmente accesible para terceras personas y mucho más sencillo que la línea de comandos.

En segundo lugar, se quiere ampliar el proyecto añadiendo otro módulo el cual implemente un algoritmo para comparar dos diagramas BPMN y podamos ver las similitudes y las diferencias. Los requisitos del trabajo son:

- El software unificará todos los módulos del proyecto.
- El software facilitará la ejecución de los diferentes módulos.
- Se debe instalar un servidor para poder abrir a terceros el programa.
- Los resultados de las comparaciones se deben mostrar de una manera que los usuarios los entiendan.

2.1.2. Posibles obstáculos

En el desarrollo del trabajo se puede tener algunos riesgos derivados del las herramientas que se usan y las personas implicadas en éste.

En primer lugar, al estar utilizando una tecnología que no se conoce a la perfección, como es JavaServer Faces, los tiempos de implementación pueden variar dependiendo de su dificultad. Esto se puede solucionar dedicándole más horas y ayudándose de toda la documentación que podamos encontrar.

También relacionado con la interfaz, uno de los aspectos más preocupantes es la pérdida de tiempo a causa de no entender el código de los otros módulos, los cuales se tienen que integrar con la interfaz y hará que tengamos que estar en contacto con todos los demás desarrolladores para aclarar

cómo funciona cada parte. A parte, una buena documentación de los otros módulos ayudará al desarrollo del trabajo.

Por último, a priori, el tiempo que se tiene para el trabajo es suficiente pero al tener múltiples factores externos a la propia implementación, como puede ser los mencionados anteriormente a parte de otros, como puede ser el tiempo de configurar el servidor donde irá la web, pueden hacer que se acorten los plazos y así tengamos menos tiempo para la implementación del nuevo módulo.

2.1.3. Metodología

Para el desarrollo del trabajo se usará una metodología, la cual consistirá en utilizar una estrategia de desarrollo incremental, con lo que conseguimos que en poco tiempo tengamos una parte funcional del software y después se irán añadiendo los diferentes módulos. También, se irán solapando algunas fases, ya que, al constar el trabajo de dos partes, las cuales no tienen una relación hasta el final, se pueden ir desarrollando en paralelo.

Todo el trabajo se desarrolla desde casa a excepción de la parte de configurar el servidor que puede ser necesario ir a donde está el servidor físico.

2.1.4. Herramientas de seguimiento

Para poder realizar el trabajo, y como es necesario tener acceso al código de los demás módulos para desarrollar parte del software, ya se está utilizando un repositorio Subversion [7] al cual tanto desarrolladores como directores tienen acceso. También es un buen sistema de seguimiento ya que se pueden ver los mensajes que se dejan en los commits y así es más fácil ver los progresos.

A parte la parte de la interfaz gráfica, una vez esté montada en un servidor accesible para todos, también se puede usar como herramienta de seguimiento para ver los avances que se van haciendo.

2.1.5. Método de evaluación

Para hacer el seguimiento del proyecto y analizar el estado de éste se ha acordado con los directores hacer reuniones semanales presenciales para que puedan ir viendo la ejecución del software e ir planeando los diferentes pasos y resolviendo dudas.

2.2. Planificación temporal

En este apartado se explicarán las diferentes tareas que se van a realizar en este proyecto y se mencionarán las diferentes alternativas al plan de acción a causa de las desviaciones puntuales.

2.2.1. Descripción de las tareas

En esta sección se describirán las tareas. La aparición de las tareas concuerda con el orden con el cual están planeadas realizarse. En el caso que no fuese así se mencionará.

La primera tarea es la investigación inicial del problema que queremos resolver. Esta tarea es la primera que se realiza y consiste en la investigación de todos los conceptos necesarios. Esto incluye, en primer lugar, la investigación de qué es un diagrama BPMN. En segundo lugar, qué tecnologías podemos utilizar para crear la interfaz y conectarla al trabajo ya realizado. Por último, se tiene que analizar en qué consiste el trabajo de los otros desarrolladores.

La realización de esta tarea consiste en estudiar la información que encontramos sobre los temas de estudio y quedar con los directores y los otros desarrolladores para que nos introduzcan sus módulos.

La siguiente tarea que se realiza es el análisis y diseño de la interfaz. El objetivo de esta tarea es analizar todos los puntos a tener en cuenta cuando vayamos a hacer la interfaz y decidir qué tipo de diseño se quiere tener en el proyecto.

Para el análisis se tendrá en cuenta la investigación inicial mencionada anteriormente, la cual nos dará una gran información de los puntos a tener en cuenta de los demás módulos para ver qué tipos de formularios necesitan implementarse o qué tipo de salidas nos dan.

En cuanto al diseño, se realizan algunos esbozos para ver cual encaja mejor con la idea del proyecto. Una vez escogida una base, ésta se usará para todo el diseño pero puede ir cambiando ligeramente en algunos puntos.

Después de tener analizada y diseñada la interfaz, el siguiente paso es implementar estas ideas. Esta tarea es la más larga que tiene el proyecto y la dividiremos en 4 partes, tres de ellas serán la implementación de las interfaces para los diferentes módulos.

1. Implementación de la interfaz del conversor de BPMN a texto: Ésta es al tarea más larga de la implementación, ya que una vez se realice ésta se puede reutilizar el código para las otras. En primer lugar, tendremos que implementar el diseño de la página, el cual se ha pensado anteriormente. Después se tiene que crear el controlador que llame al módulo correspondiente con los datos de entrada de la página.

2. Implementación de la interfaz del conversor de texto a BPMN: Con el diseño de la página ya implementado, en esta parte solo se tendrá que retocar un poco el formulario de entrada, ya que no tiene los mismos parámetros que el anterior, y crear el controlador correspondiente.

3. Implementación de la interfaz del comparador entre texto y BPMN: Esta tarea es parecida a la anterior, con la diferencia de que la interfaz será un poco diferente a la de los otros dos apartados, ya que el módulo al que llama no es un conversor sino un comparador. En cualquier caso, también se tendrá que crear el controlador que llame a la función del comparador.

4. Implementación del resto de la web: Por último, se implementará una página de inicio para la web y un sistema de login para poder ver la gente que se conecta y está interesada en el programa.

Para acabar la primera parte del trabajo, falta la configuración del servidor donde se va a instalar la aplicación. Esta tarea consiste en configurar un servidor para que se pueda acceder al programa de manera remota. Después de configurarlo se deberán hacer una serie de pruebas para verificar que todo funciona.

A partir de esta tarea empieza la segunda parte del proyecto. En primer lugar, cabe mencionar que esta parte es independiente de la anterior por lo que se puede modificar el orden entre ésta y las anteriores.

Esta tarea consiste en, junto con la ayuda de los directores, pensar e investigar uno o más algoritmos con los cuales comparar dos modelos BPMN. A parte, también se debe investigar qué tipo de comparación se quiere hacer y qué tipo de resultado se quiere obtener, ya que se pueden hacer comparaciones más estrictas o más permisivas.

Una vez hecho el análisis mencionado en el apartado anterior, se debe implementar la solución y documentarla correctamente. La implementación del módulo tendrá dos partes. En primer lugar, la propia implementación del

algoritmo o algoritmos que se hayan decidido y con todas las especificaciones que se encuentren en el análisis y todas sus restricciones. En segundo lugar, se debe implementar la parte de la interfaz que incluya el nuevo módulo y actualizar el programa en el servidor.

Para acabar, esta tarea consiste en validar, por un lado, que el nuevo módulo funciona tal y como se pedía en las especificaciones del proyecto, y su documentación es correcta. Por otra parte, se tiene que validar que el módulo se haya acoplado correctamente a la implementación de la interfaz.

2.2.2. Recursos

Los recursos que se utilizan se pueden separar en dos grupos, los recursos de hardware y los recursos de software. A continuación se listarán los recursos que se utilizarán.

A continuación hay un listado con, al principio, los recursos hardware que se utilizarán en el proyecto seguido de los recursos software que se utilizarán.

- PC: usado para el desarrollo de todo el proyecto.
- Servidor: usado para el despliegue de la web.
- Windows 10: Usado para todo el proyecto.
- Ubuntu 14.04: Usado para todo el proyecto.
- Eclipse: Usado para todo el proyecto.
- LaTeX: Usado para la memoria del proyecto.

2.2.3. Tiempo aproximado

En este apartado se marcará el tiempo aproximado que se tardará en realizar el proyecto. Para ello, se ha realizado una tabla con los tiempos aproximados y un diagrama de Gantt con todas las tareas explicadas anteriormente. Se puede asumir que le dedicamos medias jornadas, por lo que cada día equivalen a 4 horas, que nos daría un total de unas 416 horas.

Task Name	Start Date	End Date	Duration
Investigación inicial	01/02/16	10/02/16	10d
Interfaz	**11/02/16**	**06/04/16**	**56d**
Análisis y diseño de la interfaz	11/02/16	15/02/16	5d
Implementación de la interfaz	16/02/16	30/03/16	44d
Implementación del conversor de BPMN a texto	16/02/16	06/03/16	20d
Implementación del conversor de texto a BPMN	07/03/16	13/03/16	7d
Implementación del comparador entre texto y BPMN	14/03/16	20/03/16	7d
Implementación del resto de la web	21/03/16	30/03/16	10d
Configuración del servidor	31/03/16	06/04/16	7d
Modulo	**07/04/16**	**14/05/16**	**38d**
Análisis del módulo	07/04/16	16/04/16	10d
Implementación del módulo	17/04/16	06/05/16	20d
Validación del módulo	07/05/16	14/05/16	8d

Figura 2: Tabla de tiempos

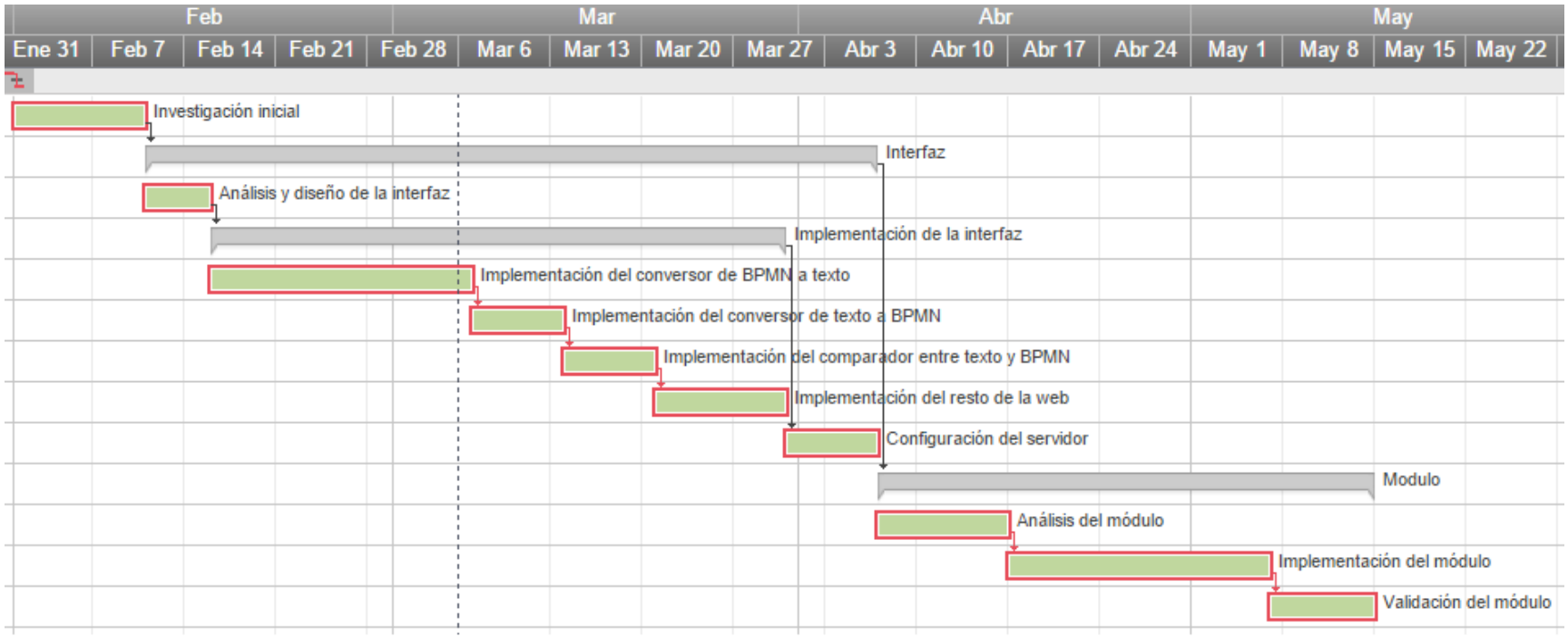


Figura 3: Diagrama de Gantt

2.2.4. Alternativas al plan de acción

Durante el desarrollo del proyecto pueden surgir diferentes desviaciones del plan original. En primer lugar, ya se han pensado en los posibles obstáculos y cómo solucionarlos.

A parte, respecto los tiempos dados, en primer lugar están en días, por lo que podemos estirar más o menos dedicando más horas al día, y en segundo lugar, tenemos un margen de unos 10-15 días para imprevistos después de la planificación y antes de la inscripción de la defensa del proyecto.

2.3. Planificación económica y sostenibilidad

En este apartado se hablará sobre el estudio de sostenibilidad que se ha hecho en relación al proyecto y la gestión económica necesaria para realizar el proyecto.

2.3.1. Gestión económica

En esta sección, se analizará y se estimará el coste total para el desarrollo del proyecto. Teniendo en cuenta que es un proyecto universitario, la parte económica por parte de los recursos humanos se contará con el salario específico de cada rol, ya que en realidad yo estaré realizando todos los roles.

Para el cálculo de la amortización hemos tenido en cuenta que el proyecto se llevará a cabo durante aproximadamente 6 meses.

En primer lugar está el presupuesto de recursos humanos. Este proyecto consta con 4 roles: jefe de proyecto, diseñador, ingeniero de software y beta tester. Con estos roles y la planificación hecha anteriormente podemos calcular el coste que tendremos que invertir para realizar el proyecto. Para saber el tiempo de cada rol se ha asignado cada tarea de la planificación en el diagrama de GANTT al rol o roles que la ejecutarían.

Rol	Horas	Precio por hora	Precio total
Jefe de proyecto	100 h	50,00/h	5.000,00
Diseñador	76 h	35,00/h	2.660,00
Ingeniero de software	188 h	35,00/h	6.580,00
Beta tester	52 h	30,00/h	1.560,00
TOTAL	416 h		15.800,00

Cuadro 1: Presupuesto en recursos humanos

Después está el presupuesto para el hardware que se va a utilizar en este proyecto.

Producto	Precio	Ciclo de vida	Amortización
PC	1.000,00	5 años	100,00
Servidor	1.500,00	5 años	150,00
TOTAL	2.500,00		250,00

Cuadro 2: Presupuesto en hardware

También tenemos el presupuesto para software. En la tabla aparece también el software gratuito para tener en cuenta todo.

Producto	Precio	Ciclo de vida	Amortización
Windows 10 Pro	150,00	3 años	25,00
Ubuntu 14.04	0,00	3 años	0,00
Wildfly 9	0,00	3 años	0,00
Eclipse	0,00	3 años	0,00
LaTeX	0,00	3 años	0,00
TOTAL	150,00		25,00

Cuadro 3: Presupuesto en software

Por último tenemos los gastos indirectos. En este proyecto los gastos indirectos que habrá son los gastos en consumo eléctrico. Para hacer el cálculo hemos buscado el consumo medio diario en diferentes fuentes y hemos hecho una media aproximada.

Producto	Precio	Cantidad	Precio total
Electricidad	0.80/kWh	2 kWh por 104 días	166.40

Cuadro 4: Gastos indirecto

Para finalizar, a continuación hay una tabla con el total del presupuesto para el proyecto.

Concepto	Coste aproximado
Recursos humanos	15.800,00
Harware	250,00
Software	25,00
Gastos indirectos	166,40
TOTAL	16.241,40

Cuadro 5: Presupuesto total

Una vez calculado el presupuesto tenemos que hacer un control de desviaciones. Considerando las especificaciones del proyecto, no será necesario más equipamiento a parte del ya especificado en los apartados anteriores por lo que por parte del hardware y del software no debería haber ningún desvío.

El mayor problema que podemos tener es que en la fase de implementación se alarguen los tiempos previstos en la planificación, lo cual hace que el presupuesto aumente. Para evitar este problema, se utilizará el diagrama de Gantt para reorganizar el tiempo para cada tarea y así ajustarse a la planificación.

En caso de que no fuese suficiente, se reservará una parte del presupuesto para contingencias. También se usará este presupuesto en el caso que el hardware que utilicemos se estropee. Esta parte equivaldría a un 10 % del total anterior por lo que el presupuesto final sería de 17.865,54.

2.3.2. Sostenibilidad social

Actualmente no existe ningún software que haga una tarea como la que se quiere realizar en este proyecto, pero lo que se va a desarrollar es una herramienta muy útil para muchas empresas que hoy en día ya utilizan diagramas BPMN pero los hacen manualmente y sin saber realmente si lo que están haciendo corresponde a lo que quieren.

Con este proyecto lo que se pretende es facilitar que las empresas desarrollen sus modelos de procesos de una manera mucho más cómoda y con un conjunto de herramientas para validar sus procesos.

2.3.3. Sostenibilidad ambiental

Por lo que respecta al área ambiental, este proyecto no tiene un gran efecto, ya que en su mayoría es un proyecto de software y en este caso lo que afecta son los gastos indirectos, en este caso la obtención de energía, la cual se hace a partir de un proceso el cual perjudica el medio ambiente.

Por otra parte, vamos a utilizar un servidor, la fabricación del cual ha dejado un impacto ambiental. En este proyecto se reutilizará un servidor que ya se estaba utilizando y de esta manera minimizamos el impacto ambiental.

2.3.4. Puntuación sostenibilidad

Con lo mencionado en los apartados anteriores, se da una puntuación para la parte inicial del proyecto teniendo en cuenta la tabla de sostenibilidad.

	Económica	Social	Ambiental
Puntuación	8	7	7

Cuadro 6: Puntuación sostenibilidad

3. Comparador de modelos BPMN

En este apartado se quiere explicar al lector en qué consiste la comparación realizada entre modelos BPMN, las limitaciones y ventajas que tiene este método respecto a otros y diferentes ejemplos y experimentos realizados para acabar de entender como funciona.

El objetivo de esta parte del proyecto es implementar un comparador de modelos BPMN con el que se consigan buenos resultados y que éstos sean entendibles por los usuarios. El problema que se plantea es un graph-matching entre dos modelos BPMN. Este módulo va dirigido, y puede ser útil, a empresas que están buscando simplificar sus procesos de negocios y quieren saber si su nuevo modelo es similar y quieren tener una comparativa rápida. Como ya se ha visto antes, ya existen maneras y técnicas para comparar dos diagramas, las cuales son lentas de ejecutar. En el caso de este trabajo la intención es integrarla con una interfaz web por lo que el tiempo de respuesta para el usuario tiene que ser rápido. Para ello se ha pensado en el uso de un algoritmo de satisfacción de restricciones, más concretamente un algoritmo Relaxation Labelling, el cual consiste en optimizar una asignación de un conjunto de variables con un conjunto de posibles valores.

La limitación que tiene este método aplicado a nuestro problema es que el algoritmo utiliza una comparativa estructural, ya que busca flujos en el diagrama parecidos. En el caso de los modelos BPMN, el contenido y contexto de las tareas es un factor importante. Se puede dar el caso que dos modelos que describen procesos sin ninguna relación sean estructuralmente parecidos o iguales y el comparador dé como resultado una equivalencia alta. En cualquier caso, este módulo tiene como premisa que si se quiere comparar dos modelos es porque en principio describen el mismo proceso.

3.1. Business Process Model and Notation

Una de las partes más importantes del proyecto y en la cual se basa todo él es la notación BPMN y más concretamente la versión BPMN 2.0 que es la que se ha utilizado. En este apartado se explicará en qué consiste y cómo está formado.

BPMN, como ya se ha mencionado antes, es una notación gráfica estándar que permite describir modelos de negocio. El formato que utiliza es un formato de flujo de trabajo (workflow) como se puede ver en la figura 4, y internamente se guarda en un formato xml.

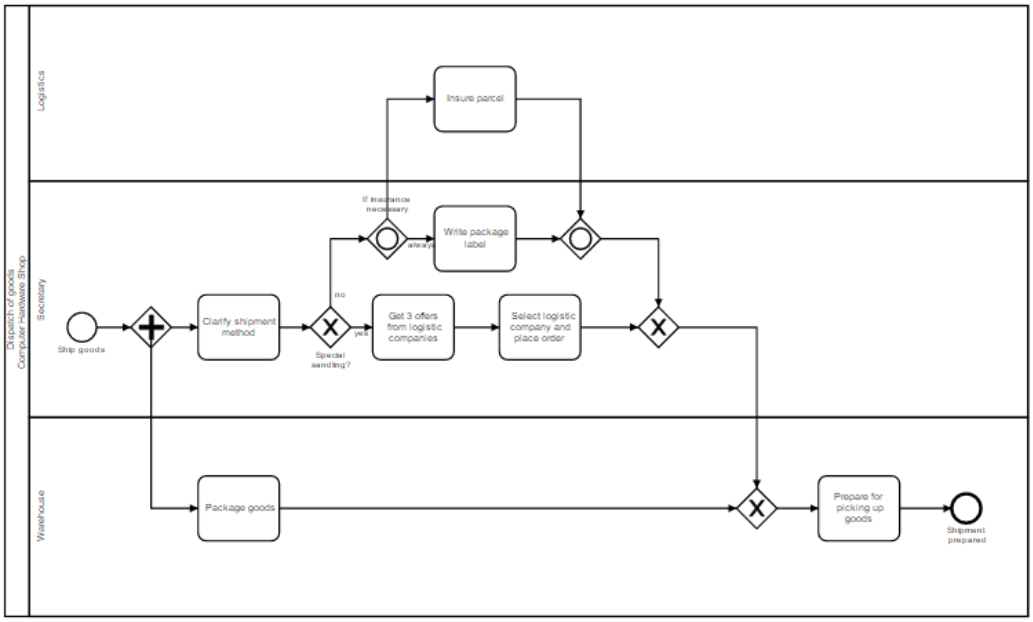


Figura 4: Ejemplo modelo BPMN

Un modelo BPMN está formado por una serie de elementos. Estos elementos se pueden clasificar dentro de 4 categorías: objetos de flujo, objetos de conexión, canales y artefactos. Más allá de lo que se explicará a continuación, algunos programas como puede ser Signavio [3] añade algunos elementos más, pero éstos ya no forman parte del estándar y algunas librerías que utilizan los módulos internamente no los soportan.

La primera categoría es objetos de flujo. Los elementos de este tipo son los elementos principales en la descripción de procesos y son nodos dentro

del diagrama. Dentro de éstos se encuentran tres tipo diferentes. Para diferenciarlos se utilizará la figura 5.

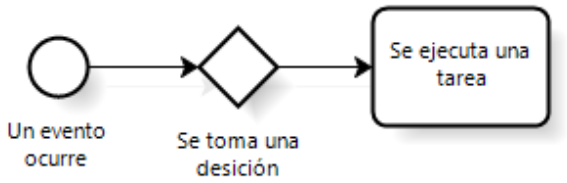


Figura 5: Tipos de Objetos de Flujo

- **Events**: Este tipo de elemento hace referencia a algún tipo de suceso en el curso del proceso, el cual puede afectar al curso de éste. Hay una gran cantidad de subtipos de eventos, por ejemplo, evento de inicio y de final, evento de recibir un mensaje, evento de que ha pasado cierto tiempo, etc. Este tipo de objeto se representa con una circunferencia como, se puede ver en el primer elemento de la figura 5.

- **Gateways**: Este tipo de elemento hace referencia a algún tipo de suceso en el curso del proceso, el cual puede afectar al curso de éste. Hay distintos subtipos de gateways, por ejemplo, gateway exclusivo, gateway inclusivo, etc. Este tipo de objeto se representa con un rombo, como se puede ver en el segundo elemento de la figura 5.

- **Activities**: Este tipo de elemento hace referencia al trabajo que se ejecuta en el proceso de negocio. Hay distintos subtipos de tareas, por ejemplo, tarea manual, tarea de usuario, etc. A parte, una tarea puede hacer referencia a un subproceso, lo que significa que dentro de esa tarea corresponde a la ejecución de otro proceso. Este tipo de objeto se representa con un rectángulo, como se puede ver en el tercer elemento de la figura 5.

La segunda categoría es objetos de conexión. Los elementos de este tipo están representados por flechas y son los que indican el flujo en la ejecución del proceso. De esta categoría hay tres tipos. En primer lugar, las líneas de secuencia que conectan objetos de flujo e indican el flujo del proceso. En segundo lugar, las líneas de mensaje que indican mensajes que se envían desde diferentes procesos de negocios y están representadas por líneas discontinuas. Por último, las líneas de asociaciones que se usan para asociar anotaciones y se representan con líneas a puntos.

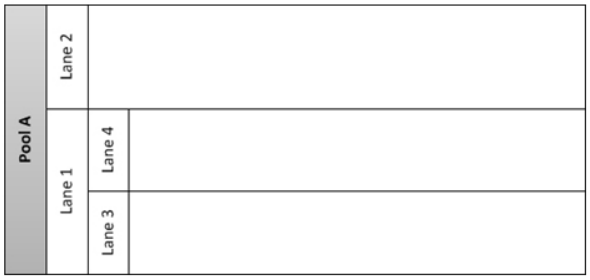


Figura 6: Tipos de Canales

La tercera categoría es canales. Este tipo de elementos sirve para organizar los diferentes flujos en diferentes categorías y se puede ver cómo son en la figura 6. De esta categoría hay dos tipos. En primer lugar, los Pools, los cuales contienen dentro un proceso de negocio, ya que en cada modelo puede haber más de uno. En segundo lugar, están las Lane, las cuales organizan y categorizan las tareas dentro de un Pool. Si el modelo solo tiene un Pool y una Lane se pueden omitir.

Por último, la cuarta categoría es artefactos. Los elementos de esta categoría sirven para añadir información adicional sobre el proceso. De esta categoría hay tres tipos diferentes. En primer lugar, objetos de datos, los cuales se representan de formas diferentes dependiendo de la procedencia de éstos y hacen referencia a archivos, bases de datos, etc. que se utilizan en el proceso. En segundo lugar, están los grupos, los cuales sirven para englobar elementos, desde tareas a Pools, a los que se les quieran dar algún significado específico, y se representa enmarcándolos en un rectángulo con línea discontinua. Por último, están las anotaciones, que sirven para describir el flujo que pueda ser dudoso.

3.2. Algoritmo de satisfacción de restricciones

El algoritmo, como ya se ha dicho anteriormente, es un algoritmo de Relaxation Labeling de restricciones, el cual consiste en optimizar la asignación de un conjunto de variables y un conjunto de posibles valores.

Sea $V = v_0, v_1, ..., v_n$ un conjunto de variables y $L = \{l_1, l_2, ..., l_m\}$ un conjunto de posibles valores para las variables. Sea $H = \{h^1, h^2, ..., h^n\}$ los pesos a optimizar donde h^i es un vector que contiene la distribución de probabilidad de v_i, tal que $h^i = \{h^i_1, h^i_2, ..., h^i_{L_i}\}$. Dado que el proceso es iterativo,

la probabilidad para el valor l en la variable v_i en el paso t es $h_l^i(t)$. Usando $R = r_1, r_2, ..., r_p$ como el conjunto de restricciones que tiene el problema donde, para la variable v_i y el valor l, $r_k(v_i, l)$ es la compatibilidad de la variable v_i y el valor l para la restricción r_k. Las restricciones pueden estar ponderadas por otras variables adjuntas. Como soporte, S_{il} representa la compatibilidad de la asignación del valor l a la variable v_i [8]. El algoritmo Relaxation Labeling lo podemos ver en la figura 7.

```
Initialize:
H := H0,

Main loop:
    repeat
    For each variable v_i
        For each posible label l for v_i
            S_il = sum_{r in R} r(v_i, l)
        End for
        For each possible label l for v_i
            h_l^i(t+1) = h_l^i(t) * (1 + S_il) / sum_{k=1}^{L_i} h_k^i(t) * (1 + S_ik)
        End for
    End for
    Until no more significant changes
```

Figura 7: Relaxation Labeling algorithm

En este apartado se explicará cómo se ha utilizado este método con el problema que se plantea, explicando las restricciones que se utilizan y el algoritmo implementado propiamente, junto las diferentes opciones de valores iniciales.

3.2.1. Restricciones

Las restricciones que se utilizan se dividen en dos grupos. Por un lado, restricciones que no dependen de otros elementos más allá de los que se están comparando. Por otro lado, restricciones que dependen de otros elementos. El valor asociado que tiene cada restricción se puede configurar.

Las restricciones que no dependen de otros elementos más allá de los que se están comparando son:

- **Restricción de tipo:** Cada tipo de elemento tiene un número de subtipos. Si los dos elementos que se comparan son del mismo subtipo se suma el valor asociado a la restricción.

- **Restricción de nombre:** Esta restricción tiene dos variantes. En primer lugar, si los dos elementos no tienen nombre se le asigna un valor. Por otro lado, si los dos tienen un nombre se separa por palabras y por cada palabra que esté en los dos elementos se le suma cierto valor.

- **Restricción de Lane:** esta restricción tiene las misma estructura que la anterior. Si los dos elementos no tienen Lane o el Lane no tiene nombre se le asigna un valor. Si los dos tienen nombre se le suma un valor por cada palabra igual.

- **Restricción del número de relaciones:** Esta restricción suma el valor asociado si los dos elementos que se están comparando tienen el mismo número de relaciones que apuntan hacia ellos con uno de diferencia, y suma el mismo valor si tienen el mismo número de relaciones que salen de ellos con uno de diferencia.

Ahora se explicarán las restricciones que dependen de otros elementos. Estas restricciones no suman un valor constante como las anteriores sino que en este caso cada restricción tiene un valor constante el cual se multiplica por las probabilidades de que otras parejas de elementos sean equivalentes. En nuestro caso estas restricciones son:

- **Restricción de elementos a un paso de distancia:** Esta restricción pondera la equivalencia de dos elementos dependiendo de si los elementos que tienen a un nivel de distancia son equivalentes. Los elementos adyacentes que se comparan tienen que ser de igual tipo ya que sino no tienen relación en la matriz de probabilidades ya que, como ya se ha dicho antes, solo se asignarán dos elementos si son del mismo tipo. También se tiene en cuenta si los elementos adyacentes tienen una flecha de entrada o de salida hacia los elementos que se están evaluando y en este caso solo se miran si los dos adyacentes son de salida o de entrada. Cuando se cumple la restricción se suma un valor, el cual se calcula multiplicando, a un valor constante elegido previamente, la probabilidad de que los dos adyacentes sean equivalentes. El valor de la probabilidad se obtiene de la matriz resultante en la iteración anterior a la actual.

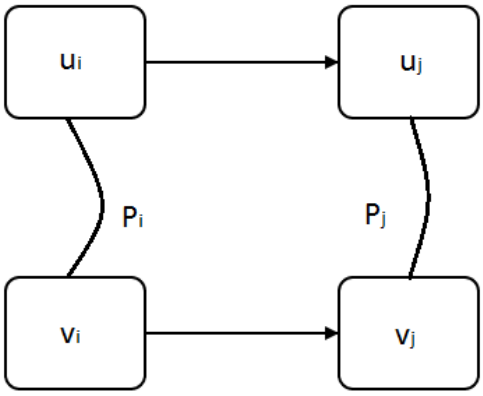


Figura 8: Restricción de elementos a un paso de distancia

$P_h(t)$ = Probabilidad de la relación h en el instante t.
$P_i(t) += constant_{valueFlow} * P_j(t-1)$

- **Restricción de elementos a dos pasos de distancia:** Esta restricción pondera la equivalencia de dos elementos dependiendo de si los elementos que tienen a dos niveles de distancia son equivalentes. Como en el caso anterior, tienen que ser del mismo tipo y se comparan los elementos de dos niveles hacia adelante entre sí y dos hacia atrás entre sí. Cuando se cumple la restricción se suma un valor, el cual se obtiene a partir de una constante elegida previamente, multiplicado por la probabilidad de los elementos a dos niveles de distancia y la probabilidad de los elementos de en medio.

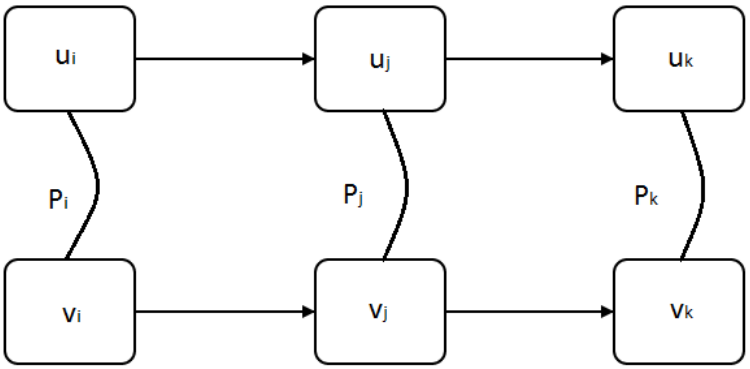


Figura 9: Restricción de elementos a dos pasos de distancia

$P_h(t)$ = Probabilidad de la relación h en el instante t.
$P_i(t) += constant_{valueDoubleFlow} * P_j(t-1) * P_k(t-1)$

- **Restricción de elementos a un paso de distancia con elementos a dos pasos de distancia:** Esta restricción pondera la equivalencia de dos elementos dependiendo de si los elementos a un nivel de distancia son equivalentes a los elementos del otro a dos niveles de distancia. Como en el caso anterior, tienen que ser del mismo tipo y el elemento del medio tiene que ser igual al siguiente, así se mira si los elementos se han separado en dos. También, como en las restricciones anteriores, se comparan los elementos de niveles hacia adelante entre sí y hacia atrás entre sí. Cuando se cumple la restricción se suma un valor, el cual se obtiene a partir de una constante elegida previamente multiplicada por la probabilidad del elemento a dos niveles de distancia con el adyacente del segundo elemento.

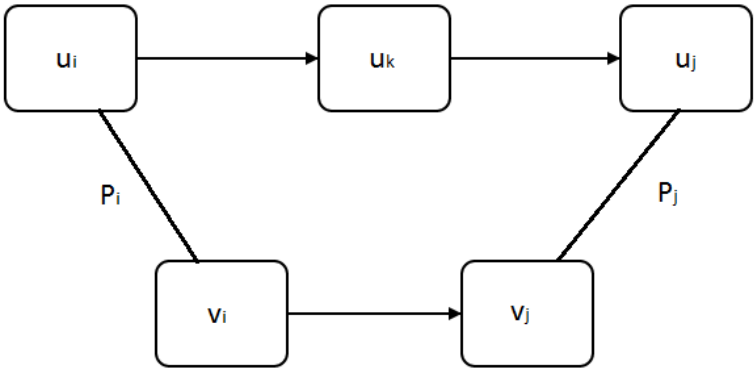


Figura 10: Restricción de elementos a un paso de distancia con elementos a dos pasos de distancia

$P_h(t)$ = Probabilidad de la relación h en el instante t.
$P_i(t) += constant_{valueOneTwoFlow} * P_j(t-1)$

- **Restricción de elementos a un paso de distancia con sentido invertido:** Esta restricción pondera la equivalencia de dos elementos dependiendo de si los elementos a un nivel de distancia en sentidos opuestos(elementos que salen de uno comparados con elementos que entran del segundo elemento). Esta restricción resta sobre el valor total, para ello la constante que se utiliza es negativa y en el algoritmo se suma para la similitud entre las restricciones del código.

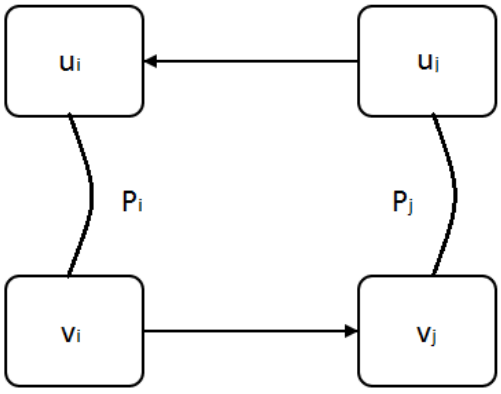


Figura 11: Restricción de elementos a un paso de distancia con sentido invertido

$P_h(t)$ = Probabilidad de la relación h en el instante t.
$P_i(t) += constant_{valueInverse} * P_j(t-1)$

3.2.2. Implementación del algoritmo

Para utilizar el algoritmo, lo primero es seleccionar los dos conjuntos de variables y posibles valores. En este caso, al programa se le pasa dos modelos que se quieren comparar estructuralmente, se coge como variables los objetos de flujo y objetos de datos, los cuales se podrían considerar nodos del diagrama, del primer modelo y, como posibles valores, los del segundo.

El siguiente paso es asignar las probabilidades iniciales. Para cada elemento del primer modelo v_i y cada elemento del segundo modelo u_j se asigna un valor inicial. En este caso se han implementado tres variantes de este valor inicial. La primera utiliza las restricciones que no dependen de ningún elemento más, la segunda consiste en utilizar valores aleatorios y la tercera trata de usar valores equivalentes para cada v_i con todos los elementos u_j.

Después llega la parte iterativa que se ha explicado anteriormente para maximizar la probabilidad. En este caso las características a tener en cuenta en cada iteración sobre las variables vecinas son las restricciones anteriores. En cada iteración se calcula un valor para cada relación v_i u_j aplicando las restricciones $R = \{r_1, r_2, ..., r_k\}$. Una vez calculados los valores se normalizan todos los valores asociados con v_i para obtener la probabilidad en esa iteración. Esto se repite hasta que converge el resultado y no hay más cambios entre los resultados de una matriz y otra, con un margen de error, o hasta cierto número de iteraciones. En la figura 12 se puede ver un pseudocódigo del algoritmo que se ha utilizado.

```
H := H_0
while(significative changes)
    for each element of model1 v_i
        for each element of model2 u_j
            S_ij = h_j^i * sum_{restriction∈R} restriction(v_i, u_j)
        end for
      h^i = normalize(S_i)
    end for
end while
```

Figura 12: Relaxation Labeling algorithm orientado al problema planteado

3.3. Resultados obtenidos

En este apartado se mostrará algunas de las pruebas y ejemplos que se han hecho a lo largo de proyecto y se explicará qué resultados se obtienen y cómo interpretar éstos. Los modelos de los ejemplos que se han utilizado se han obtenido de las pruebas de los otros módulos y de la modificación de éstas.

En los ejemplos, los parámetros constantes de las restricciones que se han utilizado, si no se dice lo contrario, son los siguientes.

- **Compatibilidad para subtipos iguales:** La constante que se utiliza cuando dos elementos tienen el mismo subtipo tiene el valor predeterminado de 35.
- **Compatibilidad para palabras iguales:** La constante que se utiliza cuando dos elementos tienen alguna palabra en común tiene el valor predeterminado de 25.
- **Compatibilidad para nombres nulos:** La constante que se utiliza cuando dos elementos no tienen nombre tiene el valor predeterminado de 10.
- **Compatibilidad de palabras de la Lane:** La constante que se utiliza cuando dos elementos tienen alguna palabra en común en la Lane tiene el valor predeterminado de 15.
- **Compatibilidad para el numero de flechas:** La constante que se utiliza cuando los dos elementos tienen el mismo número de flechas o casi el mismo de entrada o de salida tiene el valor predeterminado de 30.
- **Compatibilidad valueFlow:** La constante que se utiliza para la restricción de elementos a un paso de distancia tiene el valor predeterminado de 60.
- **Compatibilidad valueDoubleFlow:** La constante que se utiliza para la restricción de elementos a dos pasos de distancia tiene el valor predeterminado de 100 (más grande que el anterior ya que se pondera con dos probabilidades).
- **Compatibilidad valueOneTwoFlow:** La constante que se utiliza para la restricción de elementos a un paso de distancia con elementos a dos pasos de distancia tiene el valor predeterminado de 50.

- **Compatibilidad valueInverseFlow:** La constante que se utiliza para la restricción de elementos a un paso de distancia pero en sentidos opuestos tiene el valor predeterminado de -50.

Estos parámetros no tienen ninguna unidad y se han puesto como ejemplo razonable de parámetros. También decir que, como los valores que se obtienen con estas constantes se acaban normalizando, se pueden utilizar magnitudes más altas o más bajas y se obtendrían los mismo resultados. En cualquier caso, estos valores se pueden modificar antes de la ejecución.

3.3.1. Interpretación de los resultados

En este apartado se explicará qué resultados se obtienen al ejecutar el algoritmo, cómo se procesan posteriormente estos resultados y cómo se interpretan por parte del usuario. Para ver cómo funciona y que sea entendible los resultados se sacarán de la interfaz web.

El algoritmo que se ha implementado, como ya se ha mencionado anteriormente, busca una asignación de elementos entre dos modelos BPMN para ver si son equivalentes o si se asemejan. Este algoritmo intenta optimizar las probabilidades basándose en unas restricciones y completa una tabla con las probabilidades entre elementos. La figura 13 muestra un ejemplo de esta tabla correspondiente a la ejecución de la figura 14.

ID	Task_12j0pib	Task_0sl26uo	ExclusiveGateway_1mpgzhg	Task_0s79ile	Task_0jsoxba	ExclusiveGateway_0z5sib0
Task_12j0pib	0.000	0.000		0.000	0.998	
Task_0sl26uo	0.000	0.998		0.000	0.000	
ExclusiveGateway_1mpgzhg			0.999			0.000
Task_0s79ile	0.000	0.000		0.997	0.000	
Task_0jsoxba	0.000	0.000		0.000	0.998	
ExclusiveGateway_0z5sib0			0.000			0.999
InclusiveGateway_1dgb4sg			0.000			0.000
StartEvent_1						
Task_0e6hvnj	0.000	0.000		0.996	0.000	
EndEvent_1fx9yp3						
InclusiveGateway_0p2e5vq			0.000			0.000
ExclusiveGateway_1ouv9kf			0.000			0.000
ParallelGateway_02fgrfq			0.000			0.000
Task_0vaxgaa	0.000	0.000		0.000	0.000	
Task_05ftug5	0.000	0.000		0.000	0.000	

Figura 13: Tabla de resultados

En la tabla de la figura 13 se puede ver en el eje vertical los elementos del primer modelo que se le pasa al algoritmo identificados por los ids interno del

modelo, el único dato único que se puede usar para diferenciar elementos, y en el eje horizontal aparecen los elementos del segundo modelo también identificados por los ids. Cada celda muestra la probabilidad obtenida entre los dos elementos que corresponde. Si los dos modelos son parecidos, la mayoría de las celdas deberían de tener una probabilidad de 0 (o vacías en el caso que los elementos sean de tipos diferentes y por lo tanto no se han comparado entre ellos) y las que sean equivalentes tendrían un valor cercano a 1.

En el caso que no haya una relación clara entre un elemento del primer modelo con alguno del segundo, esto provocará que en la fila de ese elemento se vea más de un valor y la suma de éstos será 1. Esto significa que no tiene una relación 100 % segura y que podría ser equivalente a uno de los elementos que tengan valor diferente a 0.

Aunque éste es el resultado real del algoritmo, es un resultado poco intuitivo para alguien que no conoce cómo funciona el programa y le puede costar entender si sus modelos son parecidos o no. Para ello, después de obtener la matriz de probabilidades, se ejecuta un post-procesado de ésta, el cual escoge para cada elemento del primer modelo el elemento del segundo modelo con el que tenga una relación con mayor probabilidad y que sea mayor a una variable que se puede configurar, para no aceptar elementos con poca similitud. Este post-procesado hace que cada elemento del primer modelo solo pueda apuntar a un elemento del segundo, pero cada elemento del segundo puede ser apuntado por más de uno del primero. En la interfaz se interpreta estos datos y se obtiene resultados como los de la figura 14.

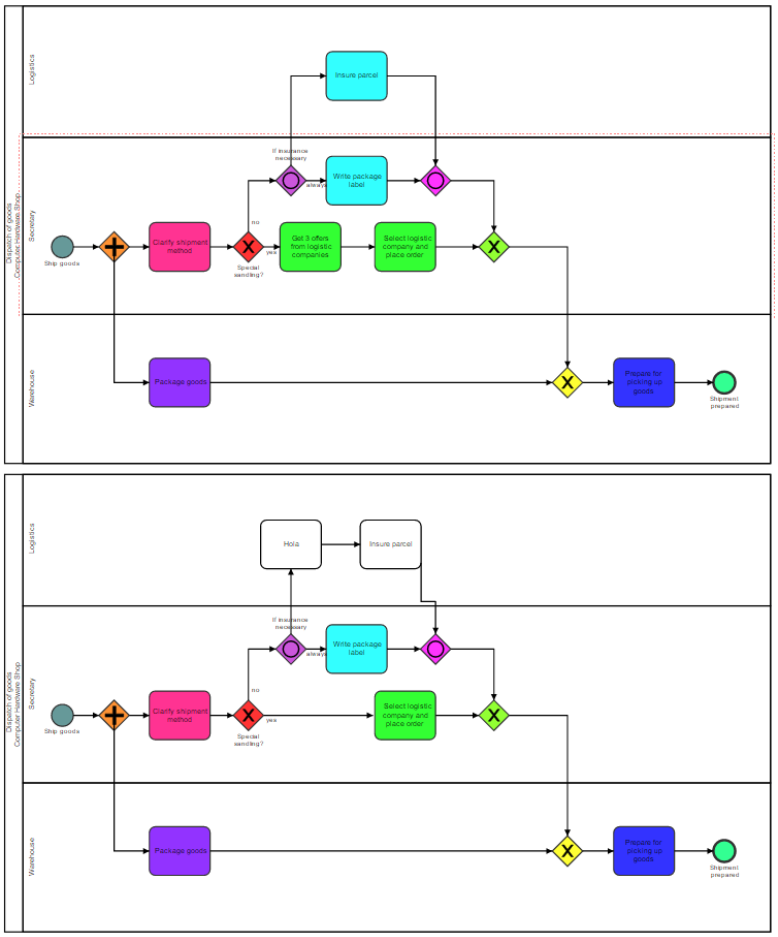


Figura 14: Ejemplo de resultados

En la interfaz se puede ver cómo se asignan las relaciones entre los dos modelos con un código de colores. Así se puede ver rápidamente qué elementos son equivalentes ya que éstos tendrán el mismo color.

Por último, como el post-procesado explicado antes cambia dependiendo del orden en que se le pasen los modelos, se ha pensado en proporcionar un valor de similitud absoluto a los dos modelos. Para ello, se ejecuta dos veces el algoritmo cambiando el orden de los modelos. Con los resultados del post-procesado de cada ejecución se comparan entre ellos. En primer lugar, se cuentan las relaciones de la primera ejecución que también están en la segunda ($Igual_1$) y las que no ($noIgual_1$). Después se hace lo mismo pero con las de la segunda ejecución con las de la primera para obtener $Igual_2$ y $noIgual_2$. Finalmente, se devuelven tres valores que se obtienen de la siguiente manera:

$result = (Igual_1 + Igual_2)/(Igual_1 + noIgual_1 + Igual_2 + noIgual_2)$
$result_{Norm} = (Igual_1)/(Igual_1 + noIgual_1)$
$result_{Inverse} = (Igual_2)/(Igual_2 + noIgual_2)$

El primer valor se puede interpretar como una igualdad entre los dos modelos. Con los dos valores siguientes se consigue más información de esta igualdad.

3.3.2. Comparar modelos iguales

Las primeras comparaciones que se han realizado han sido entre modelos iguales. En estos test se ha querido comprobar que claramente el resultado que se obtiene es que cada elemento se corresponde a sí mismo en el otro modelo. Efectivamente, el resultado que se obtiene es el deseado y el valor obtenido de equivalencia es de 100 %. La figura 15 muestra un ejemplo de esta ejecución.

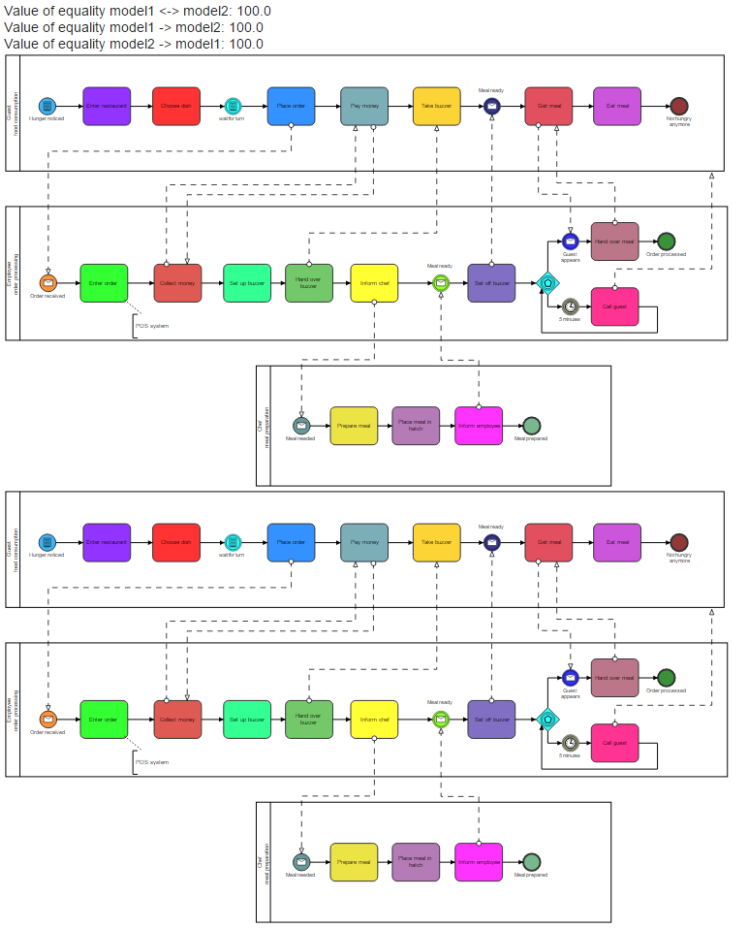


Figura 15: Ejecución modelos iguales

El siguiente paso de estos test fue ir modificando varios valores de las constantes de las restricciones multiplicándolos hasta por 100 y los resultados que se obtuvieron eran iguales. Esto es lo esperado ya que, aunque se aumente las constantes, el elemento más parecido es el mismo, por lo que seguirá teniendo las mismas características.

A parte de aumentar los valores, también se probó disminuyéndolos y en este caso se encontraron resultados diferentes. Si se disminuye poco a poco las variables sin ser negativas el resultado se mantenía, pero en el momento que pasan a ser negativas(o positiva en el caso de la del flujo inverso) el algoritmo empieza a dar resultados erróneos como de la figura 16.

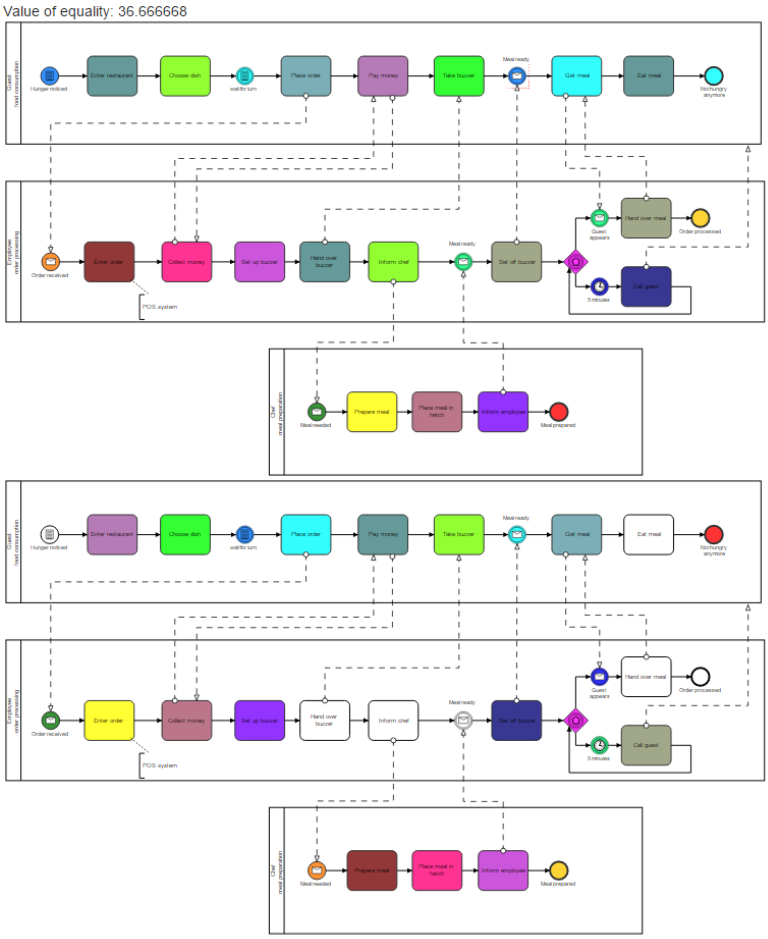


Figura 16: Ejecución con constantes negativas

En este ejemplo se cambió el valor de la constante de palabras iguales por -25 y el del nombre de lineas por -15. Esto es el resultado esperado, ya que, si se invierte el signo de las constantes se interpreta como que las características que se parecen entre los elementos perjudican en el valor de igualdad entre los elementos.

3.3.3. Comparar modelos parecidos

Las siguientes comparaciones son entre dos modelos que tienen muchos elementos en común pero no son iguales. Para ello, se cogió los ejemplos que se estaban utilizando y se modificaron dividiendo tareas para especificar más flujo y se agruparon tareas para que fuesen más genéricas.

Para estas comparaciones el resultado gráfico depende del orden en el que se le pasen los modelos por lo que se ha ejecutado dos veces cada uno. Los resultados obtenidos se pueden ver en la figura 17.

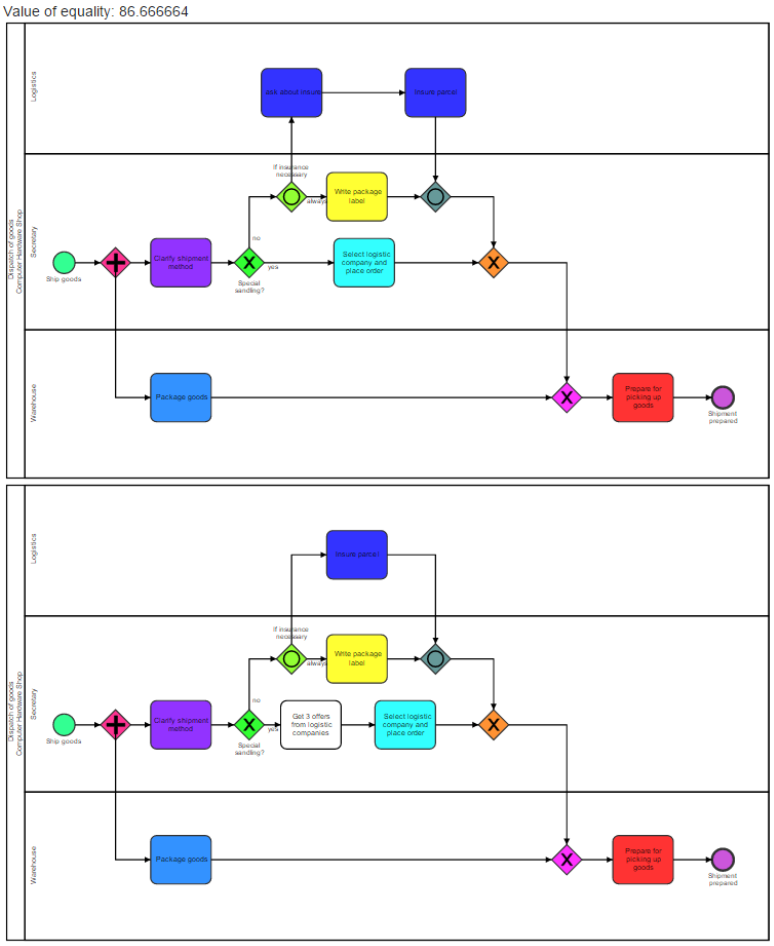


Figura 17: Primera ejecución modelos parecidos

En la primera ejecución se puede ver que se obtiene el resultado esperado. La parte que en el modelo de arriba tiene un solo elemento y en el de abajo tiene dos se ha asignado correctamente (solo se asigna una de las dos tareas porque de la manera que esta hecho cada elemento del modelo superior apunta a un elemento del modelo inferior pero un elemento del modelo inferior puede ser apuntado por más de uno del superior). Después hay una parte donde en el modelo de arriba tiene dos elementos y en el de abajo solo

uno. En este caso sí que se asignan tres elementos con el mismo color como se puede ver en la figura 18.

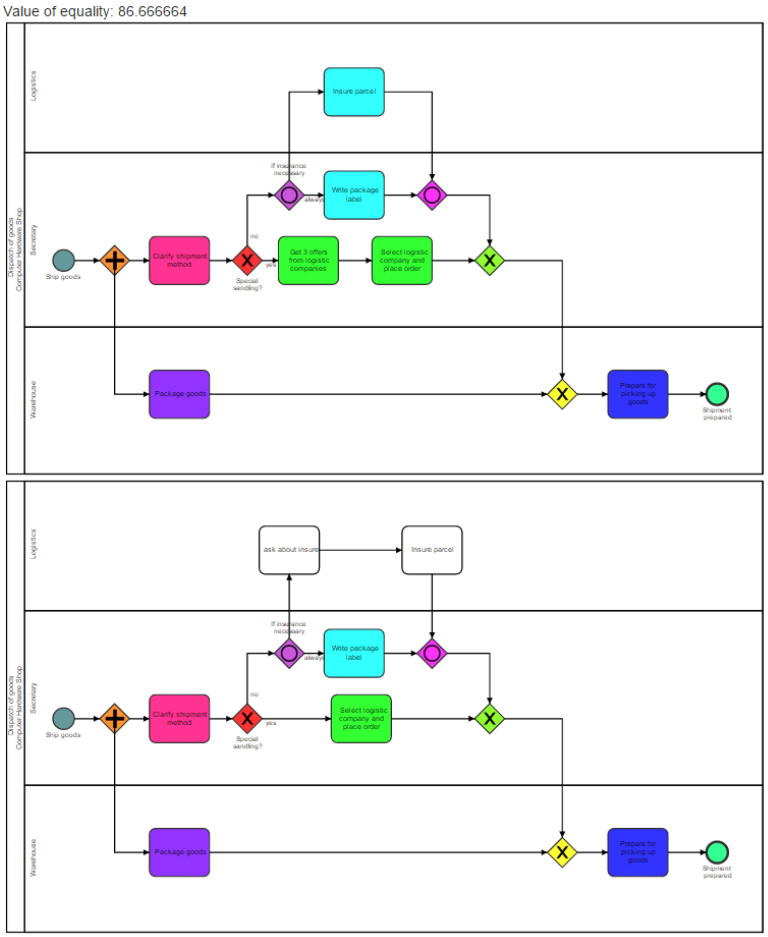


Figura 18: Segunda ejecución modelos parecidos

Usando este último ejemplo se puede ver claramente cómo actúan las constantes de las restricciones. La parte superior de los modelos ha sido modificada para que uno tenga dos elementos y el otro solo uno. Lo normal sería que uno de los dos elementos se asignará al del modelo (dos elementos del modelo inferior no pueden apuntar al superior), pero lo que hace es asignar el elemento que está en el modelo que solo hay una tarea con otro elemento que tiene como antecesor y predecesor los mismo elementos. Esto es porque las constantes le dan más importancia al flujo que a los nombres, que en este caso es igual.

Para ver cómo afecta se cambia el valor de la constante de la restricción de palabras iguales y se le pone un valor de 100. Cuando se ejecuta se consigue el resultado de la figura 19.

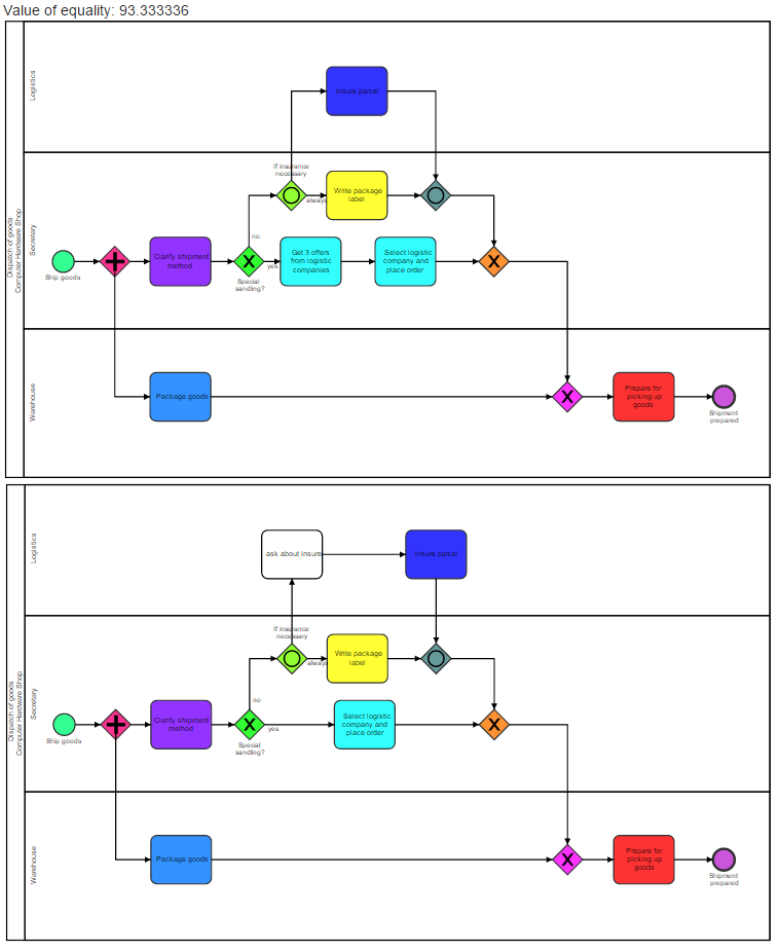


Figura 19: Comparación modelos parecidos cambiando valores

Se puede ver como ahora sí que se muestra el resultado esperado y también ha aumentado el valor de igualdad de 86,66 a 93,33.

3.3.4. Comparar modelos totalmente diferentes

Por último, también se comparan modelos que no tenían nada que ver entre ellos para ver el tipo de resultados que nos daban. En este caso, solo

se verá una de las ejecuciones, ya que los resultados ya nos indican que son muy diferentes. Ésta corresponde a la figura 20.

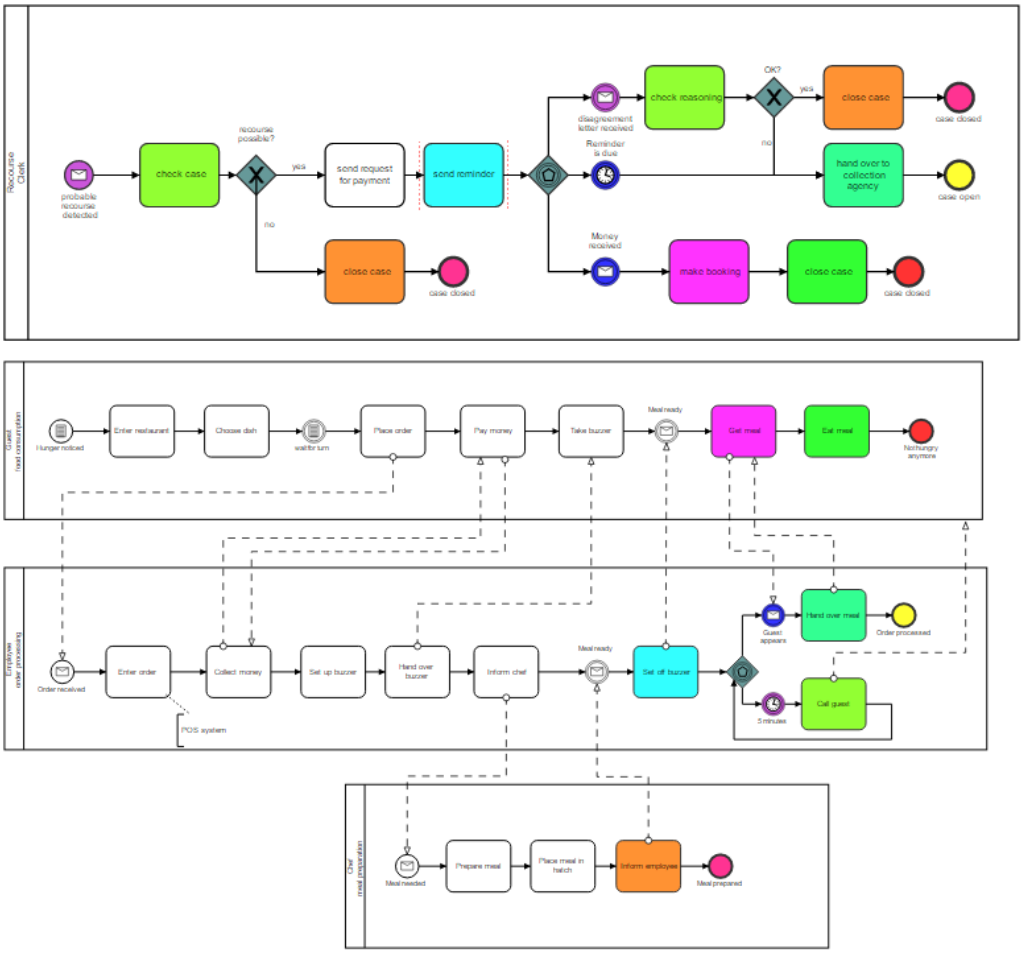


Figura 20: Comparación modelos diferentes

En esta comparación el valor de equivalencia es de 16.66, lo cual ya nos indica que no se parecen nada. Lo que se ve en la parte gráfica es que el comparador iguala estructuras parecidas pero, en este caso, solo utiliza la parte final de los elementos al asignar los eventos finales de los dos modelos y a partir de ahí encuentra estructuras parecidas. Se pueden diferenciar dos partes.

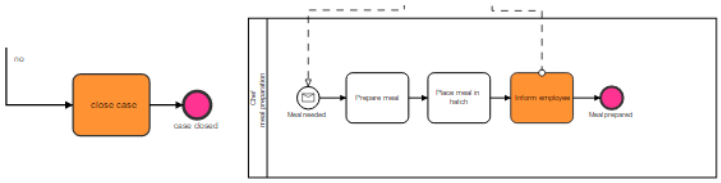


Figura 21: Primera parte igual

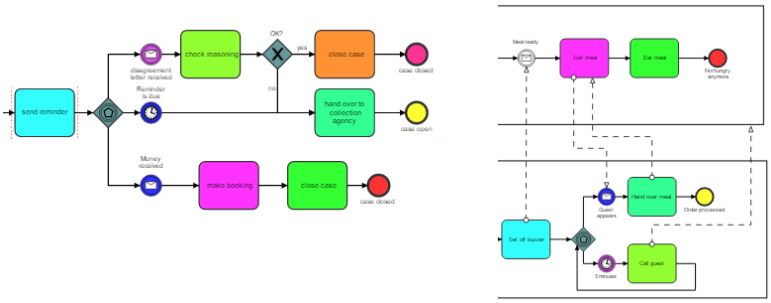


Figura 22: Segunda parte igual

La primera estructura, que corresponde a la figura 21, es básicamente una tarea seguida del evento final que ha quedado suelto. La segunda, que corresponde a la figura 22, se puede ver que se han mezclado dos estructuras más complejas. En primer lugar, todo empieza con una tarea que está pintada de color azul flojo y a partir de ese punto los dos se dividen en unas estructuras parecidas. También se puede ver que en el segundo modelo hay un bucle el cual provoca que en la asignación se muestre una asignación duplicada que sería la de los dos gateways (los elementos en forma de rombo).

3.3.5. Diferencia entre valores iniciales

Ahora se explicará las diferencias en el uso de las distintas asignaciones iniciales. En primer lugar están las dos asignaciones con las que se consiguen resultados iguales que son la informada y la equivalente. La primera, como ya se ha comentado anteriormente, consiste en aplicar las restricciones que no necesitan valores previos. La segunda consiste en asignar equitativamente las probabilidades. La diferencia entre las dos solo se puede apreciar internamente y es que, como la primera ya implica una información relevante inicial, tarda menos iteraciones en encontrar la solución en comparación con la segunda.

Después está el random, el cual, en los casos de comparar iguales o casi iguales, hay una alta probabilidad de obtener los mismos resultados que con los otros dos, pero puede pasar que se asigne una probabilidad muy baja al inicio, casi 0, y no pueda encontrar la misma asignación. En el caso que no se parezcan sí que se verán resultados más dispares ya que el valor inicial en estos casos influye más.

3.4. Experimento con ejemplos reales

En este apartado se va a realizar una serie de pruebas del módulo en las cuales se van a usar los ejercicios realizados en un seminario. Estos ejercicios consistían en crear un modelo BPMN con unas especificaciones. En este experimento se compararán los modelos de los alumnos con el del profesor y se verá si se parecen. Estos modelos se han obtenido del proyecto del comparador de texto con modelos BPMN y los modelos han sido modificados ligeramente ya que utilizaban una extensión del lenguaje que no era compatible con las librerías utilizadas en este proyecto.

El experimento consta de dos partes. En primer lugar, se seleccionan unos cuantos ejercicios realizados por los alumnos y se pasan por el algoritmo. Con esto se obtienen los tres resultados de equivalencia. Una vez obtenidos todos se calcula la media de los resultados y su desviación estándar para ver como se comportan los resultados. En segundo lugar, una vez tenemos la desviación estándar se extraen los modelos que estén fuera del rango en los tres valores y se estudian visualmente si realmente son modelos poco parecidos o porqué dan esos resultados.

El experimento se ha realizado con el modelo del profesor que se puede ver en la figura 23 y los resultados son los que indica la siguiente tabla.

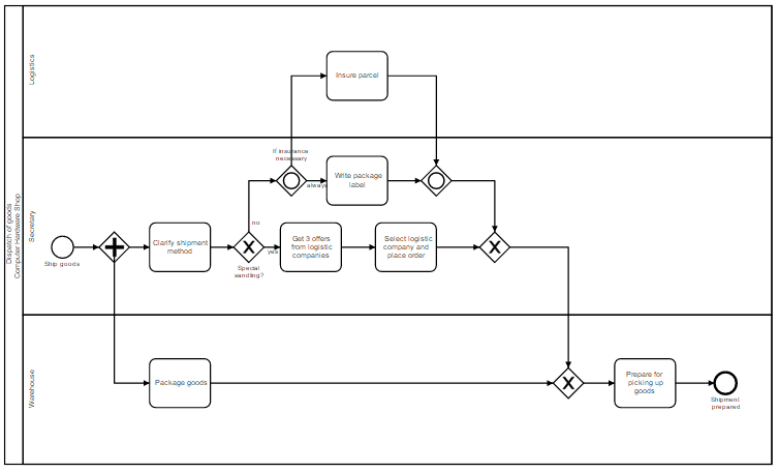


Figura 23: Ejercicio del profesor

	modelo1 $\leftrightarrow$ *modelo*2	modelo1 $\rightarrow$ *modelo*2	modelo2 $\rightarrow$ *modelo*1
Promedio	57.03	65.04	51.66
Desviación estándar	12.12	14.81	12.34

Cuadro 7: Resultado estadísticos del experimento

Con estos resultado se ha buscado qué modelo tenia los peores valores y estaba fuera de la desviación estándar. Se ha encontrado con el modelo de la figura 24. Analizando la estructura se puede ver que hay bastantes diferencias, como por ejemplo, al gateway final se llega por tres caminos en vez de por dos como el original, hay pocas tareas en comparación, lo que indica que está menos especificado.

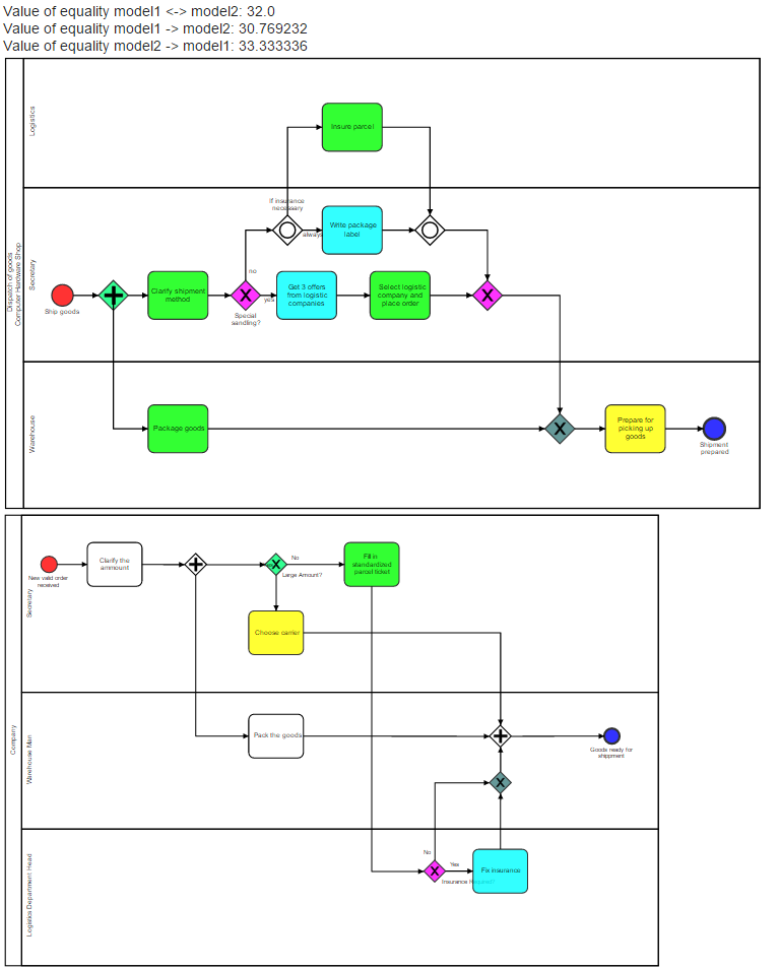


Figura 24: Peor modelo del experimento

En comparación, en la figura 25 se puede ver un ejemplo que ha obtenido un buen resultado. Se ha invertido el orden de los modelos en la ejecución solo para obtener el mejor resultado gráfico. Este ejemplo no es el que tiene las puntuaciones más altas, pero es uno de los que tiene los tres valores bastante próximos y con valores altos.

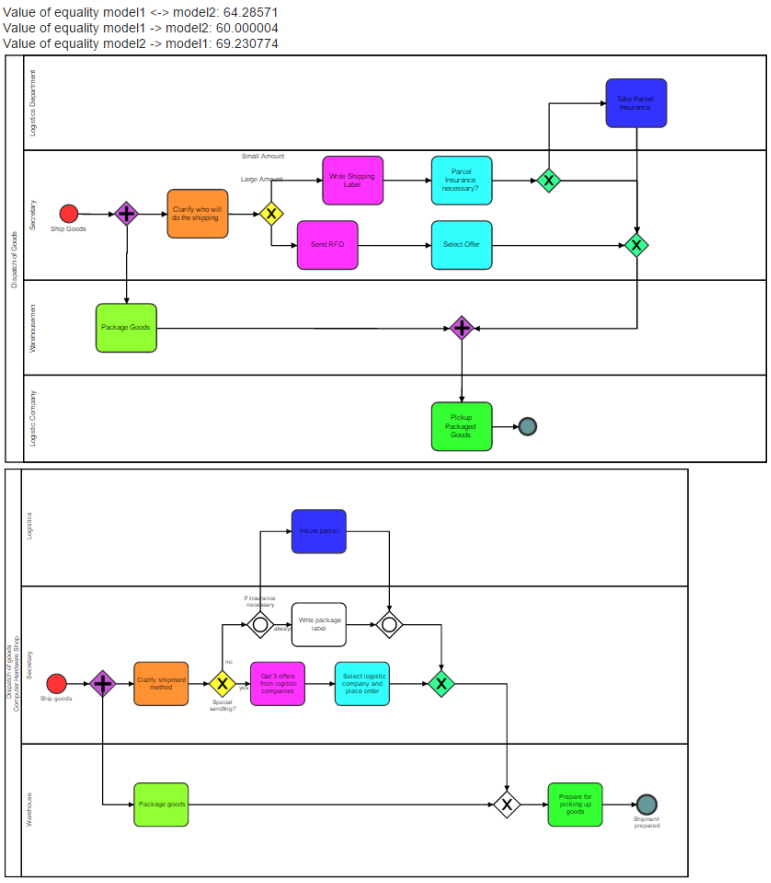


Figura 25: Buen modelo del experimento

4. Interfaz web

En este apartado se explicará cómo se ha desarrollado la interfaz web que se ha creado en este proyecto. Se hablará de en qué consiste cada módulo que se ha integrado en la web, cómo se han integrado, que más tiene la web a parte de los módulos y cómo se ha configurado el servidor para la aplicación.

Esta parte del proyecto tiene como objetivo integrar en una interfaz los cuatro módulos que se han implementado hasta el momento, contando con el explicado en el apartado anterior, y representar las entradas y los resultados de éstos. La aplicación que se explica en este apartado tiene como requisitos que sea fácil de utilizar e intuitiva, que los resultados sean entendibles y que sea accesible para los usuarios. Para ello se ha optado por una interfaz web desplegada en un servidor al cual tenga un acceso público.

Para las funcionalidades de la aplicación se ha pensado en un sistema en el cual todos los módulos comparten información. En la aplicación hay un sistema en el cual los usuarios pueden subir y descargar sus modelos o textos para tenerlos guardados, a parte se dan unos de ejemplo para probar la aplicación. Una vez subidos, los ficheros son accesibles a todos los módulos en los que se pueden utilizar. Cuando un usuario ejecuta uno de los módulos de conversión el resultado se almacena en su carpeta y ya lo tiene accesible en los otros módulos. Se puede ver un esquema de las funcionalidades en la figura 26.

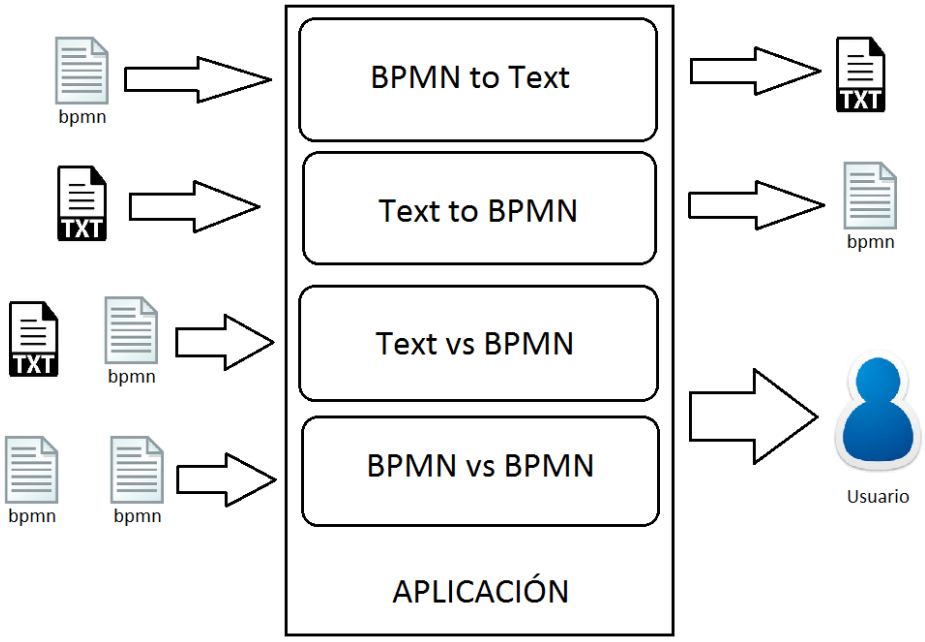


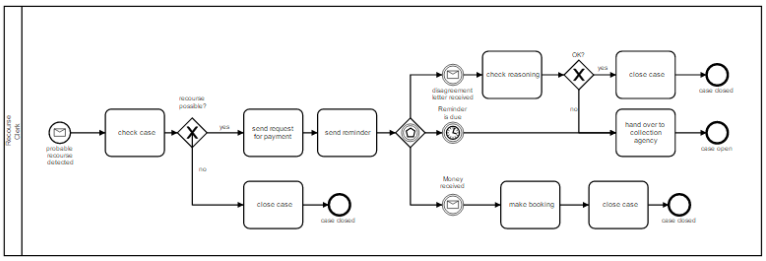
Figura 26: Esquema funcional

Antes de nada mencionar que los módulos que se explicarán a continuación se han realizado en otros trabajos finales de grado menos el comparador de modelos que ya se ha explicado anteriormente.

4.1. Módulos del proyecto

A continuación se explicarán cómo funcionan los diferentes módulos y qué opciones configurables tienen.

En primer lugar se creó un conversor el cual permite pasar un modelo BPMN a un texto escrito automáticamente. Este módulo ha sido creado por Genís Martín en su trabajo final de grado. Este módulo sirve para tener una explicación de las diferentes etapas del proceso de negocio en un escrito, lo cual es más entendible para los usuarios pero no para las máquinas. Para ello se utiliza un archivo de reglas para formar las frases del texto. El programa tiene diferentes parámetros de ejecución que se pueden elegir. Se puede elegir si el escrito es en inglés o en catalán, hay tres opciones de formatos de salida a elegir, también se puede elegir si se quiere que se separen por comas y si se quiere que se tabulen las lineas. El resultado es el texto que crea a partir del modelo. Un resultado de este proceso se puede ver en la figura 27.




```
About recourse
clerk, starting with recourse
clerk receives a message from .It continues with one of these two options will be selected:
    - The first alternative, first, recourse
clerk close case, and then one of the next 2 things has to be done:
        * If the case is recourse possible? under value ,at least one these 2 groups of activities has to be done:
            - The group of things to do in group 1, recourse
clerk check case.
            - The option 2 with condition yes, recourse
clerk send request for payment.
        * In case the condition recourse possible? is ,starting with recourse
clerk send reminder. It ends with recourse
clerk check reasoning.
    - The second alternative, recourse
clerk close case.
Finishing with recourse
clerk hand over to collection agency.
```

Figura 27: Ejemplo conversor BPMN to Text

El segundo módulo consiste en lo contrario que el anterior, permite pasar de un texto a un modelo BPMN. Este módulo ha sido creado por Didac Martínez y Arnau Gil entre sus dos trabajos finales de grado. Este módulo nos sirve para describir en lenguaje normal un proceso de negocio y convertirlo en un modelo BPMN el cual puede ejecutar una máquina. El programa permite elegir entre inglés, catalán y castellano para los ficheros de texto de entrada. El resultado es el modelo que crea a partir del texto y se puede ver un ejemplo en la figura 28.

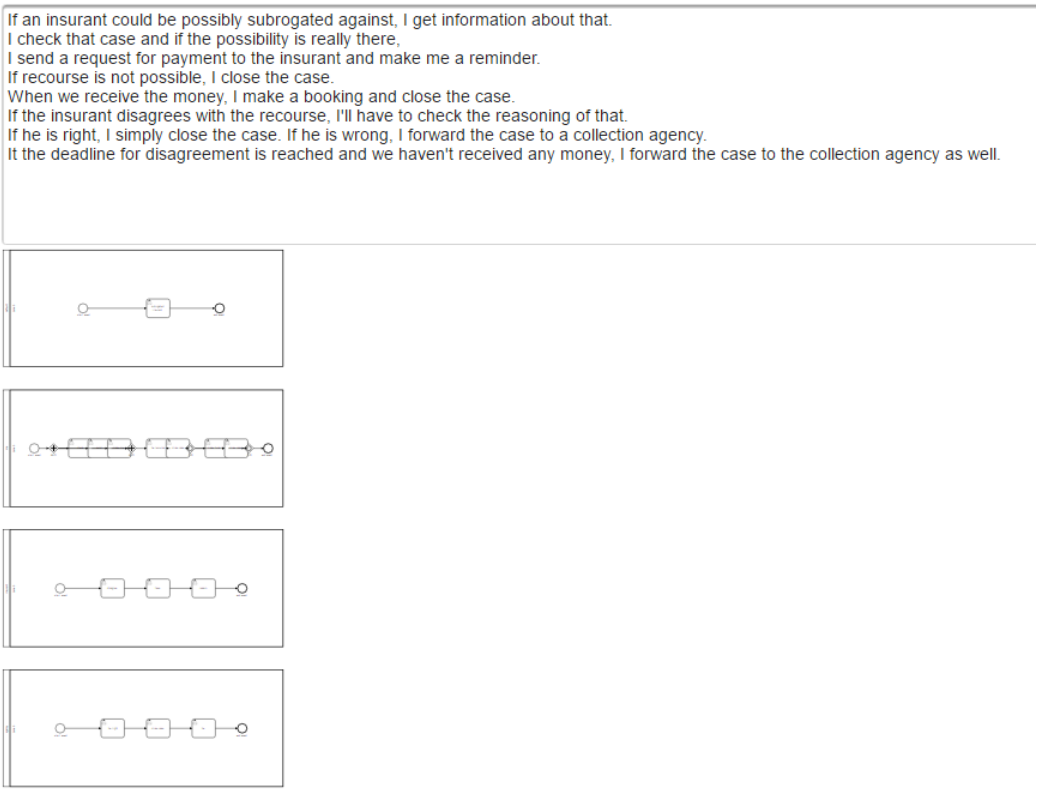


Figura 28: Ejemplo conversor Text to BPMN

El resultado que se obtiene inicialmente no es bueno visualmente ya que se solapan las tareas pero se genera un segundo archivo donde se mejora este aspecto aunque elimina información importante. Este segundo archivo corresponde a la figura 29.

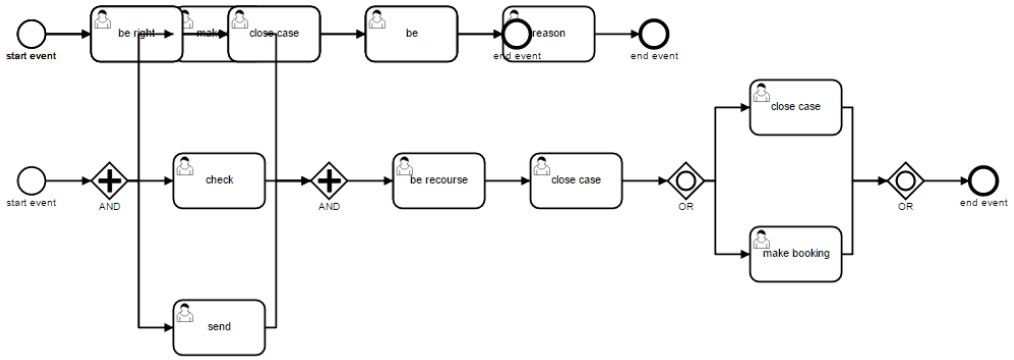


Figura 29: Ejemplo conversor Text to BPMN mejorado

El tercer módulo consiste en un comparador al cual se le pasas un modelo BPMN y un texto y devuelve su parecido a nivel de proceso. Este módulo ha sido creado por Josep Sanchez en su trabajo final de grado. Este programa no tiene ningún parámetro configurable para su ejecución en estos momentos. El resultado que se obtiene es, por un lado, la asignación de cada frase del texto con las tareas correspondientes del diagrama. Junto a la asignación se dan un listado con los motivos de esa asignación para ver si se ha asignado porque tienen muchas características comunes o pocas. También se obtiene un valor del 0 al 1 de lo parecido que son el texto y el modelo. Un ejemplo de esta ejecución se puede ver en la figura 30.

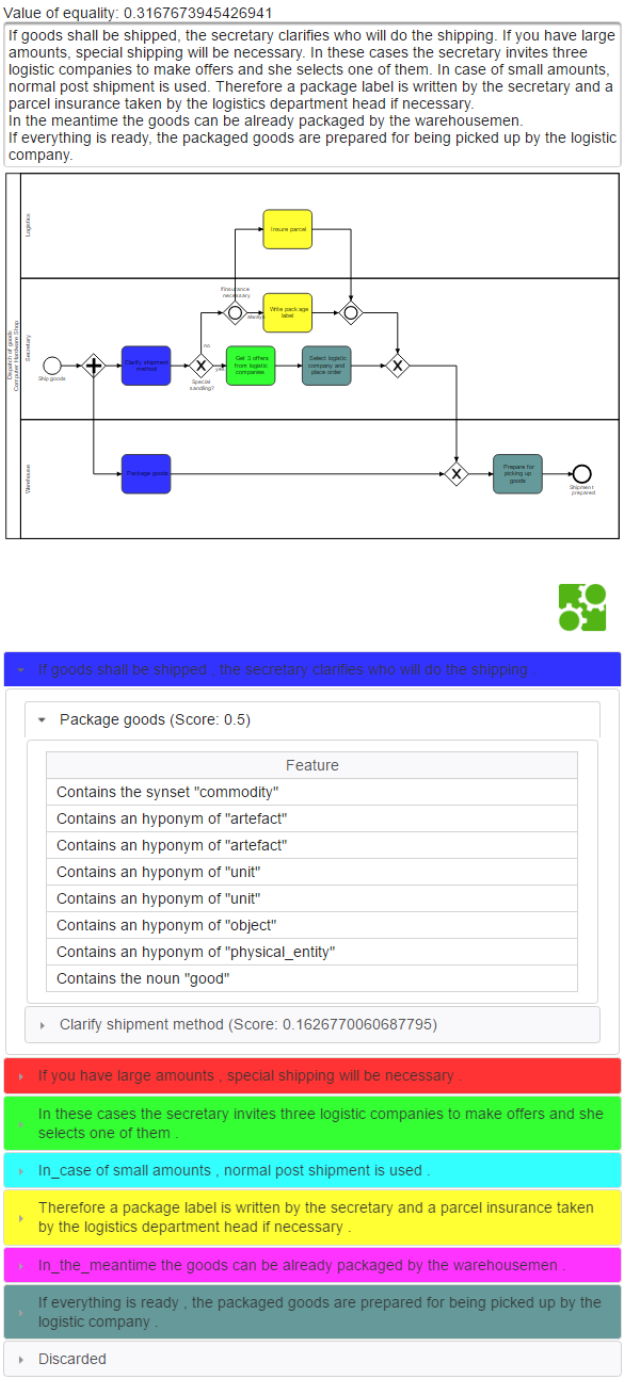


Figura 30: Ejemplo comparador Text vs BPMN

Por último, el módulo que se ha creado en este trabajo final de grado junto a la interfaz, el cual ya se ha explicado antes.

4.2. Integración de los módulos

Las pantallas que se han creado para cada módulo tienen el mismo patrón. En la parte superior hay un menú para ir a las diferentes páginas. En la parte de la izquierda hay una columna donde se encuentran los formularios para los parámetros de configuración de cada módulo, unos elementos para poder subir nuestros modelos o textos a la web y el botón de ejecución. A parte, en algunas pantallas son necesarios algunos elementos más que ya se comentarán específicamente más adelante. En la parte derecha de la pantalla se encuentran uno o dos listados, dependiendo de la pantalla, en los cuales se puede seleccionar los textos y los modelos que se han subido al servidor y unos cuantos de prueba para ver cómo funcionan los módulos. Finalmente, en la parte central se encuentran la combinación de dos elementos diferentes que son o un display para ver modelos BPMN o un cuadro de texto para los textos. El display que se utiliza es la API de JavaScript bpmn-js la cual la proporciona bpmn.io, uno de los editores online de BPMN 2.0 y utiliza las librerías de Camunda para la visualización. [9].

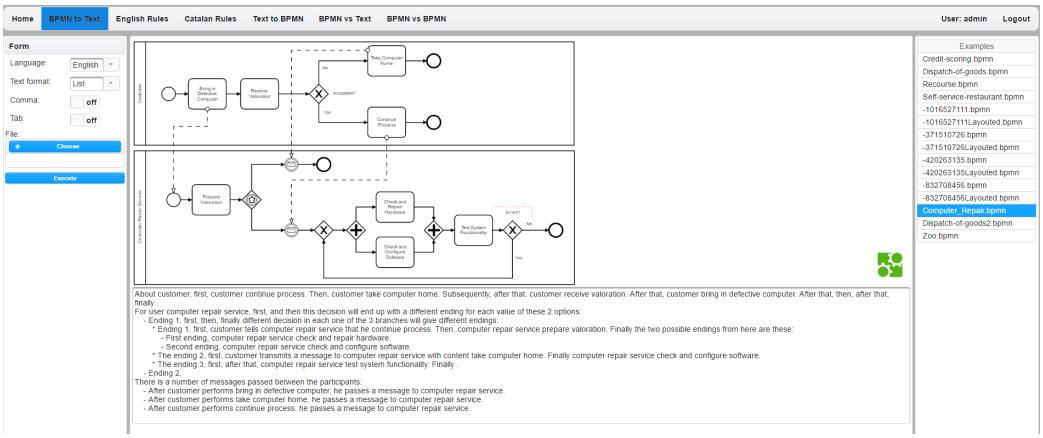


Figura 31: Pantalla BPMN to Text

La primera pantalla, que se puede ver en la figura 31, es la pantalla del módulo conversor de modelos BPMN a textos. Esta pantalla contiene solo una lista en la parte derecha con los modelos BPMN, ya que como input solo admite esto. En la parte central se encuentra en primer lugar un render del bpmn seleccionado y debajo de esté un recuadro donde se mostrará el resultado de la ejecución.

A parte, este módulo cuenta con otras dos pantallas solo accesibles si el usuario tiene el rol de administrador. Las pantallas, que se puede ver en la figura 32, están hechas para poder cambiar los archivos de reglas que necesita la aplicación. En las pantallas hay, a parte del menú inicial, un cuadro de texto en el que se ve todo el archivo de reglas y un botón de guardar. Estas pantallas solo son accesibles para administradores, ya que al guardar se tocan los archivos de reglas de la aplicación y si se guardan reglas erróneas el módulo dejaría de funcionar para todos los usuarios.

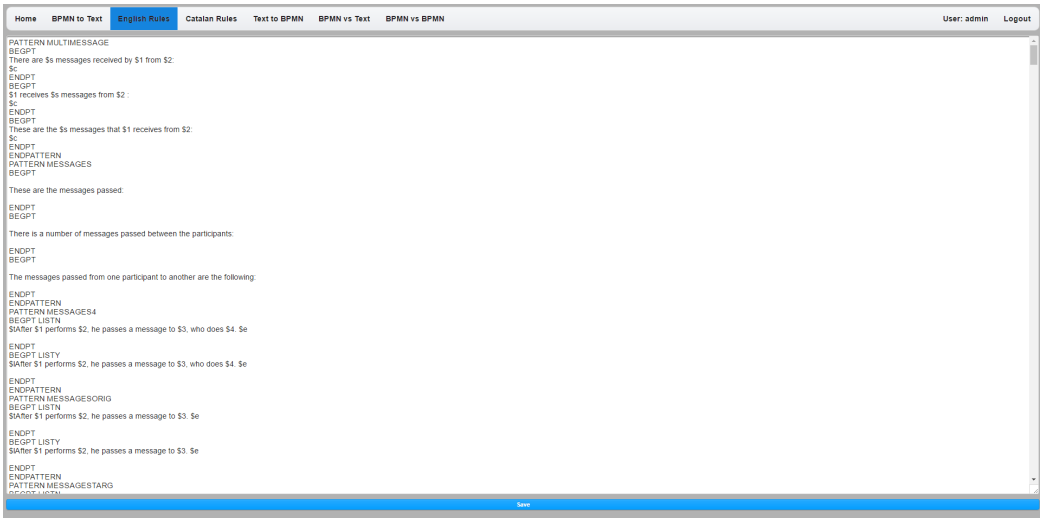


Figura 32: Pantalla reglas BPMN to Text

La segunda pantalla, que se puede ver en la figura 33, es la pantalla del módulo conversor de textos a modelos BPMN. Esta pantalla, al igual que la anterior, tiene solo una lista en la parte derecha, en este caso contiene los archivos de los textos, ya que solo admite textos como input. En la parte central se encuentra en primer lugar un cuadro de texto donde se muestra el texto seleccionado, se puede escribir uno o modificar. Debajo de éste un lugar para el render del BPMN de la solución.

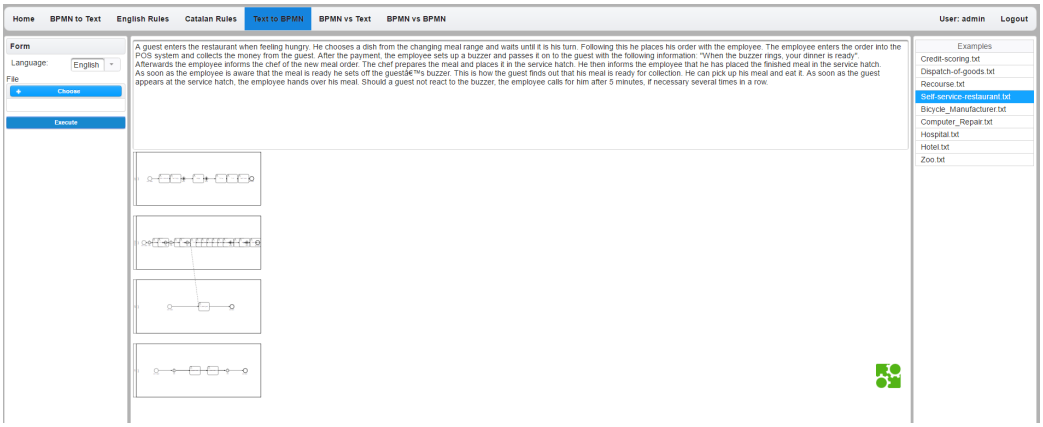


Figura 33: Pantalla Text to BPMN

Cuando se ejecuta el módulo genera dos modelos, uno completo y otro con una mejor distribución en la parte visual pero se eliminan los canales. En el render se muestra el primero pero se guardan los dos en la carpeta del usuario. Se puede ver la diferencia en la figura 34.

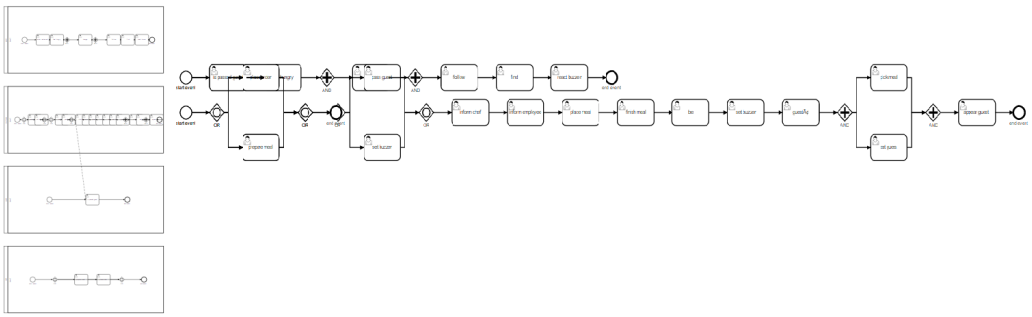


Figura 34: Diferencia entre modelos obtenidos

La tercera pantalla, que se puede ver en la figura 35, es la pantalla del módulo comparador entre textos y modelos BPMN. Esta pantalla tiene dos listas en la parte derecha. La primera contiene el listado de textos y la segunda contiene el listado de modelos, así se eligen los dos elementos que se le pasa al módulo. En la parte central se encuentra en primer lugar un cuadro de texto donde se muestra el texto seleccionado o escrito. Debajo de éste hay una parte donde se muestra el diagrama seleccionado.

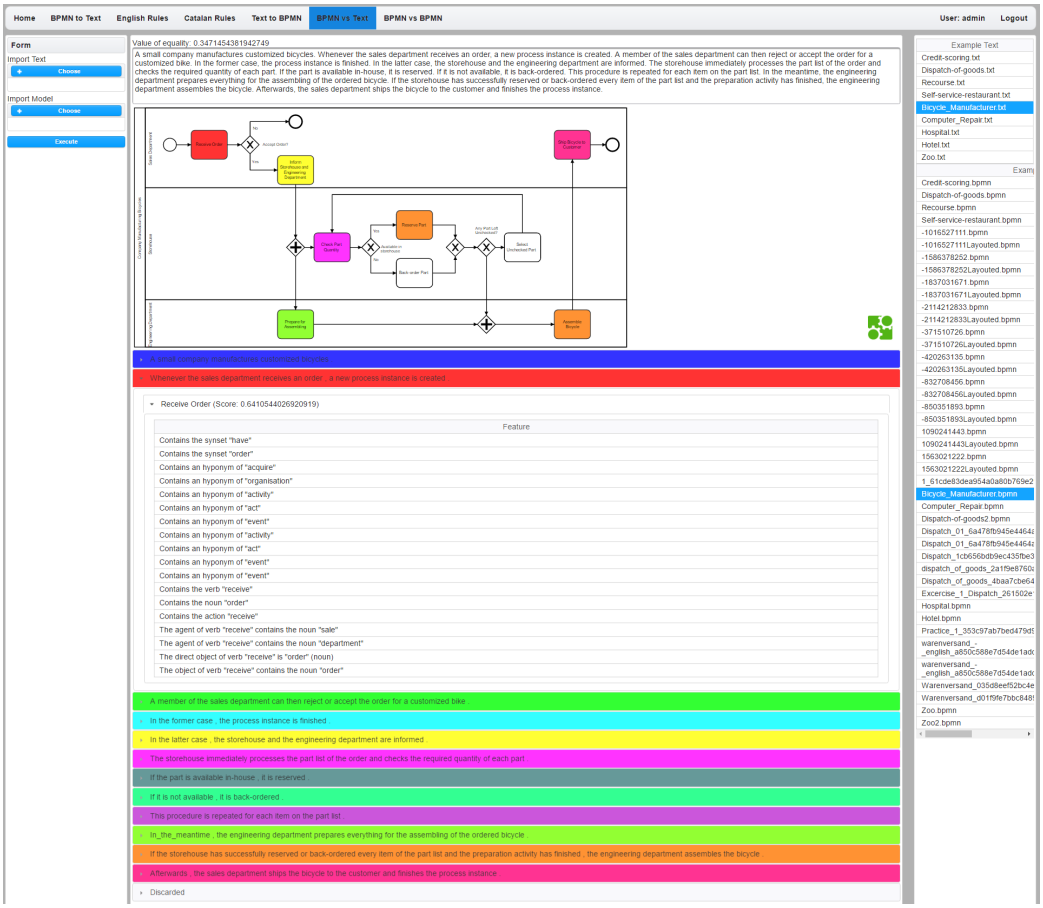


Figura 35: Pantalla Texto vs BPMN

Una vez ejecutado el algoritmo aparece, en primer lugar, un valor encima del cuadro de texto con el resultado numérico del algoritmo y, en segundo lugar, un panel en el cual se puede ver cada frase del texto y si se pulsa en una frase se ve un listado de las tareas asociadas a la frase y los motivos por los que se asocia. A parte, cada frase del panel se le asocia un color y en el render del modelo se muestra cada elemento asociado a la frase del mismo color que ésta, de esta manera es fácil interpretar los resultados.

Por último, la cuarta pantalla, que se puede ver en la figura 36, es la pantalla del módulo comparador entre dos modelos BPMN. Esta pantalla, como en la anterior, tiene dos listas en la parte derecha. En este caso las dos listas contienen el listado de los modelos que tiene acceso el usuario, así se elige los dos modelos que se van a comparar. En la parte central hay dos zonas donde se renderizan los dos modelos seleccionados.

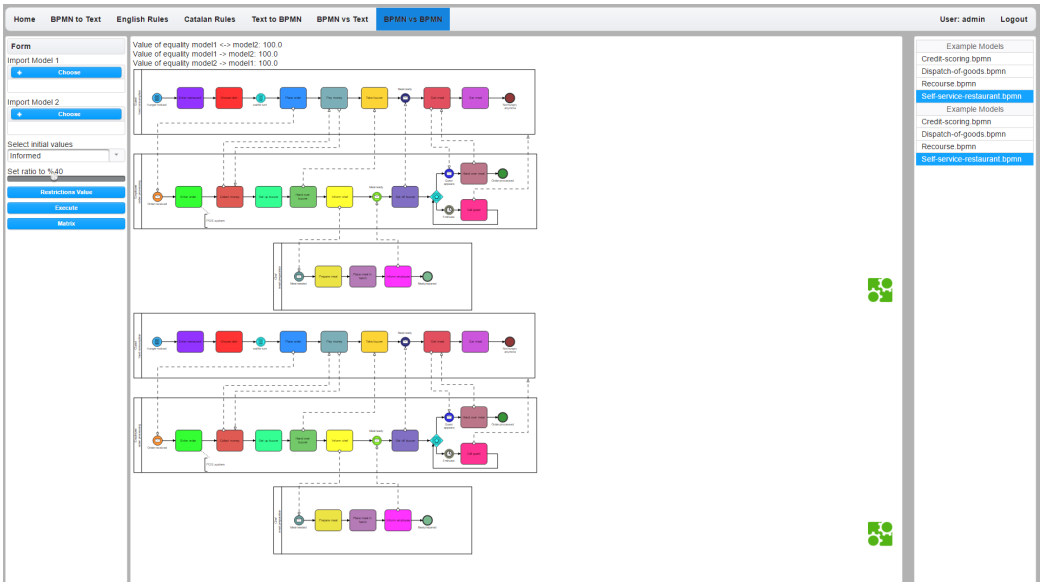


Figura 36: Pantalla BPMN vs BPMN

Una vez ejecutado el algoritmo, en primer lugar, aparece en la parte superior los tres valores que devuelve el algoritmo y que se han explicado anteriormente, en segundo lugar, los elementos equivalentes de los modelos aparecen del mismo color para ver rápidamente la relación y, por último, aparece un botón debajo del botón de ejecutar en la parte derecha de la pantalla el cual permite visualizar la matriz de resultados, como se puede ver en la figura 37.

ID	Task_12j0pib	Task_0sl26uo	ExclusiveGateway_1mpgzhg	Task_0s79ile	Task_0jsoxba	ExclusiveGateway_0z5sib0
Task_12j0pib	0.000	0.000		0.000	0.998	
Task_0sl26uo	0.000	0.998		0.000	0.000	
ExclusiveGateway_1mpgzhg			0.999			0.000
Task_0s79ile	0.000	0.000		0.997	0.000	
Task_0jsoxba	0.000	0.000		0.000	0.998	
ExclusiveGateway_0z5sib0			0.000			0.999
InclusiveGateway_1dgb4sg			0.000			0.000
StartEvent_1						
Task_0e6hvnj	0.000	0.000		0.996	0.000	
EndEvent_1fx9yp3						
InclusiveGateway_0p2e5vq			0.000			0.000
ExclusiveGateway_1ouv9kf			0.000			0.000
ParallelGateway_02fgrfq			0.000			0.000
Task_0vaxgaa	0.000	0.000		0.000	0.000	
Task_05ftug5	0.000	0.000		0.000	0.000	

Figura 37: Tabla de resultados

4.3. Otros aspectos de la web

A parte de la ejecución de los módulos, la aplicación cuenta con más funcionalidades para facilitar la tarea al usuario y mejorar la experiencia. A continuación se explicaran estas funcionalidades.

En primer lugar, la aplicación cuenta con un sistema de login y de registro para personalizar el contenido de la interfaz dependiendo del rol que desarrolle y guardar los archivos subidos por separado para que otros usuarios no los puedan ver. Para administrar las sesiones la aplicación utiliza Java Authentication and Authoritation Server (JAAS) el cual consiste en que el propio servidor de aplicaciones es el encargado de comprobar que el usuario existe y de saber qué roles tiene. En la figura 38 se puede ver los dos formularios que se utilizan para el sistema. En la parte del registro se ha puesto un captcha para evitar entradas de bots. Internamente la información que se guarda del usuario es: nombre de usuario, contraseña encriptada, email y el rol que tiene dentro de la aplicación. Los roles que puede desempeñar un usuario son dos. En primer lugar, el rol 'Usuario' el cual se asigna por defecto en el registro y permite el uso de los diferentes módulos. En segundo lugar, el rol 'Administrador' el cual se tiene que asignar manualmente en la base de datos y permite al usuario entrar en las páginas de modificación de reglas.

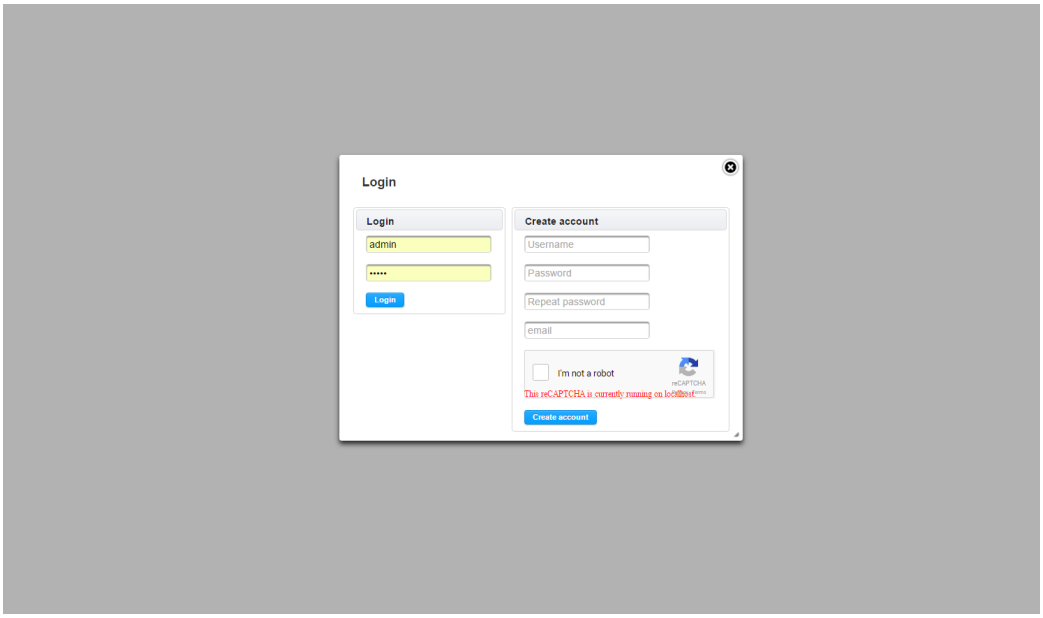


Figura 38: Pantalla de login

Una vez se entra con un usuario a la aplicación, la primera pantalla que aparece es la pantalla home. Esta pantalla sirve como gestor para los archivos del usuario. Tiene dos listas con los modelos BPMN y los textos accesibles por el usuario conectado los cuales, si se pulsa en uno se mostrará en la parte central. A parte, si se pulsa con clic derecho en un archivo de las listas aparece un menú con el cual el usuario puede eliminar sus archivos y descargarlos. En el caso que intente eliminar un archivo de los de ejemplo aparece un aviso de que no es posible. Esta pantalla con el menú corresponde a la imagen 39.

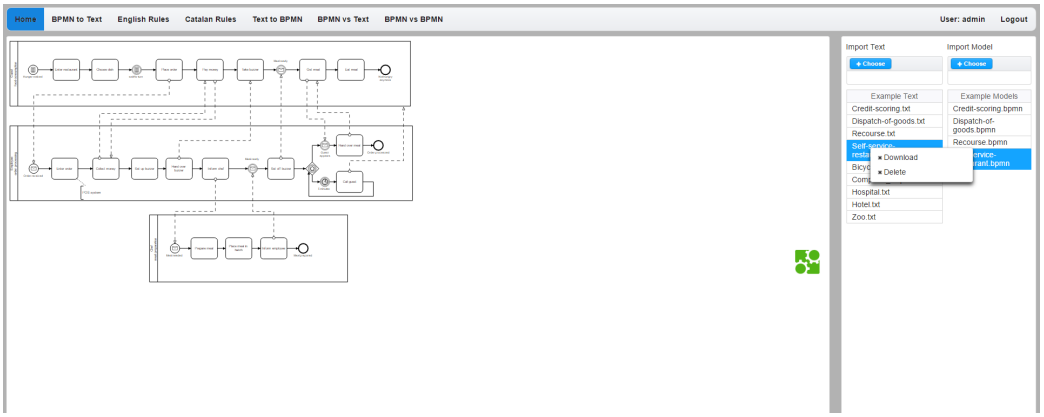


Figura 39: Home de la aplicación

4.4. Tecnologías integradas

Más allá de incluir dentro de la aplicación los códigos de los diferentes proyectos que han formado los módulos, éstos también utilizan tecnologías que se han tenido que incorporar al proyecto. En este apartado se explicará todas las tecnologías que se utilizan en la aplicación, ya sean propias de ésta o necesarias para la ejecución de uno de los módulos.

- **JavaServer Faces(JSF)**: JSF es un framework de Java el cual se utiliza para la creación de interfaces web y simplifica el desarrollo de aplicaciones. Actualmente este framekwork es un estándar por la comunidad Java. En el proyecto se ha utilizado esta tecnología para el desarrollo de la web junto al lenguaje HTML, compatible con el framework [10].

- **Wildfly**: Wildfly es el servidor de aplicaciones que se utiliza en el proyecto. Este servidor esta desarrollado por Red Hat y implementa Java Plataform, Enterprise Edition (Java EE). En el caso de este proyecto se esta trabajando con la ultima versión hasta el momento que es Wildfly 10 [11].

- **MySQL**: Esta tecnología es una base de datos desarrollada por Oracle la cual se puede integrar bien con Java y el servidor de aplicaciones que se utiliza en el proyecto. Ésta solo se utiliza para guardar los datos de los usuarios [12].

- **Petrify**: Esta herramienta sirve para la síntesis de Petri nets con un control asíncrono. Esta herramienta se utiliza internamente en el algo-

ritmo del conversor de BPMN a texto y en la aplicación únicamente se ha tenido que configurar los parámetros correctamente [13].

- **lpsolver**: Esta librería nos permite resolver Mixed Integer Linear Programming. Se utiliza internamente en el algoritmo del comparador de textos y modelos BPMN [14].
- **WordNet**: WordNet es una base de datos léxica en la cual se agrupan palabras en sinónimos. Esta tecnología también se utiliza en el comparador de textos y modelos BPMN [15].
- **Activiti**: Activiti es una plataforma para BPM. Esta plataforma contiene un core de procesado de BPMN 2.0 el cual se puede integrar en aplicaciones Java. En el proyecto se ha utilizado de dos maneras. En primer lugar, internamente para leer los modelos y manipularlos se utiliza la librería que nos proporciona. En segundo lugar, usando el plugin del editor Eclipse, se ha utilizado para modelar ejemplos y comprobar resultados [16].
- **bpmn.io**: El editor bpmn.io creado por Camunda es un editor simple el cual podemos utilizar tanto online como nativo. A parte de usar el editor para modelar junto a Activiti, este editor contiene una API que nos permite poner un viewport en una pagina web y renderizar un modelo BPMN. De esta manera se han renderizado los modelos en la aplicación [1].

4.5. Puesta en marcha del servidor

Por último, una vez realizada la aplicación se tiene que instalar en un servidor. En este caso, y como ya se ha comentado en apartados anteriores, se ha instalado en un máquina del departamento y con un servidor de aplicaciones Wildfly 10. Para poder cambiar el servidor en un futuro se ha creado un manual de instalación en el cual se describe paso por paso qué es lo que se tiene que hacer para desplegar una aplicación en una nueva máquina.

5. Conclusión

Como se ha dicho al principio de este documento, la notación BPMN se ha extendido en el mundo de los negocios y hoy en día una gran cantidad de compañías la utiliza para la descripción de sus procesos de negocio. Gracias a este hecho han aparecido una serie de herramientas para ayudar en la gestión de este tipo de diagramas.

Después de realizar el proyecto se pueden extraer varias conclusiones. En primer lugar, el hecho de que en este proyecto se hayan integrado una serie de módulos los cuales proporcionan a los usuarios unas herramientas, inexistentes en el mercado hasta el momento, se puede pensar que hay un gran número de funcionalidades útiles que aún no existen en el mercado. En segundo lugar, y entrando más en el proyecto, aunque los modelos BPMN no tratan únicamente aspectos estructurales sino que también cuenta mucho el contexto del proceso, suponiendo que dos modelos tienen un contexto similar una equivalencia estructural es un buen método para por lo menos saber si el nuevo modelo va por buen camino. Por último, aunque el hecho de depender de los navegadores en la realización de una interfaz puede limitar algunos aspectos, el hacer una interfaz web, no solo en este proyecto sino en todo proyecto el cual se esté ejecutando ya sea por línea de comandos o en una interfaz nativa, facilita muchas cosas, desde el mantenimiento entre multiplataforma, el hecho de poder ejecutar desde cualquier sitio o simplemente, en el caso que antes fuese por línea de comandos, la facilidad en reejecutar y sacar resultados entendibles.

5.1. Trabajo de futuro

Este proyecto tiene muchos aspectos en los cuales se puede seguir trabajando, tanto en la parte web como en el módulo del comparador.

- En el proyecto se ha conseguido realizar una interfaz útil para el usuario que quiere ejecutar los diferentes módulos. A partir de aquí, y una vez empiecen a utilizarlo los usuarios se podrán sacar más ideas para mejorar la experiencia del usuario dependiendo de sus necesidades.
- La aplicación que se ha realizado es una aplicación funcional. Un aspecto a seguir trabajando es el aspecto visual el cual se podría mejorar.

- En cuanto al servidor, se ha integrado en una máquina propia. La siguiente fase en este aspecto podría ser una migración a algún sistema en la nube, el cual, aunque se tenga que invertir algo de dinero, facilitaría el mantenimiento del equipo y de la aplicación.
- En el algoritmo, en primer lugar se podría aumentar el número de restricciones aunque las que se utilizan actualmente ya están dando buenos resultados. También se podría cambiar la granularidad de la comparación, ya que, el algoritmo se basa en la comparación de elementos del modelo pero se podría aumentar y pensar en comparar estructuras internas como si fuesen nodos.
- Por último, se podría implementar algún otro tipo de algoritmo para compararlo con el ya realizado.

5.2. Planificación temporal y económica final

En cuanto a la planificación temporal se tenía previsto tener finalizado el proyecto a mediados de mayo. Finalmente, después de algunos problemas que ya se dijeron que podían suceder, como los cambios en algunos de los otros módulos que provocaron el cambio en alguna pantalla de la interfaz, el proyecto se acabase a principios de junio.

Cabe decir que la mayor parte del proyecto ya estaba terminada para las fechas acordadas y la mayor parte de lo que se hizo después fue añadir utilidades para el usuario, no solo en la interfaz, sino también en el algoritmo cuando se dejó modificar las constantes de las restricciones y se añadió más tipos de asignaciones iniciales. Por último, se cambió la planificación de la configuración del servidor final para que cuando se hiciese ya tuviese el programa final y no se tuviese que acceder al servidor constantemente ya que no tenía acceso a él.

Por parte de la planificación económica no ha cambiado nada respecto a lo calculado al principio. Como ya se había mencionado, los resultados anteriores son teniendo en cuenta que se realiza en una empresa, pero que en realidad la mano de obra no supone ningún coste.

A. Manual de instalación

En este anexo se explica cómo configurar el entorno, tanto servidor como base de datos, para la aplicación web que se ha realizado. Esta configuración se basará en la instalación usando como sistema operativo Ubuntu.

A.1. Instalación de la base de datos

En el proyecto se utiliza una base de datos MySQL por lo que nos tendremos que instalar la misma. Para hacerlo podemos utilizar el comando:

```
sudo apt-get install mysql-server
```

Después, tenemos que crear un schema con el nombre de bpmninterface, y una vez creado se crea la tabla que guardará los usuarios que se registran. La query para crear la tabla es la siguiente:

```
CREATE TABLE `bpmninterface`.`user` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `username` VARCHAR(100) NOT NULL,
  `password` VARCHAR(100) NOT NULL,
  `email` VARCHAR(100) NOT NULL,
  `rol` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`id`));
```

Con esto ya tendremos todo lo necesario por parte de la base de datos y solo nos haría falta entrar cuando queramos hacer que un usuario sea administrador dentro de nuestra aplicación. Para hacerlo tendremos que cambiar el campo rol del usuario y ponerlo con el valor 'Administrador'.

A.2. Instalación del servidor Wildfly 10

Para instalar el servidor se puede seguir la siguiente guía con el único cambio de en vez de usar el archivo standalone-full.xml se utiliza el standalone.xml por lo que se debe cambiar cuando aparezca. También decir que no es necesario crear un usuario para la aplicación si no se quiere y solo la va a administrar una persona.

https://gesker.wordpress.com/2016/02/09/wildfly-10-on-ubuntu-15-10/

A.3. Cambios en el servidor

Una vez tengamos instalado el servidor y configurado el usuario de este tendremos que hacer unos cambios en el servidor. Lo primero es acceder al archivo standalone.xml que esta en $WILDFLYPATH/standalon/-configuration. Dentro del archivo tenemos que añadir, en la sección de datasources donde hay uno de ejemplo, las siguientes lineas cambiando el usuario y el password de la base de datos y poniendo el puerto correspondiente en la url si es necesario ya que el 3306 es el que viene por defecto.

```
<datasource jndi-name="java:/mysql" pool-name="mysqlDS"
    enabled="true" use-java-context="true">
    <connection-url>jdbc:mysql://localhost:3306/
        bpmninterface</connection-url>
    <driver>mysqlDriver</driver>
    <transaction-isolation>TRANSACTION_READ_COMMITTED</
        transaction-isolation>
    <pool>
        <min-pool-size>10</min-pool-size>
        <max-pool-size>100</max-pool-size>
        <prefill>true</prefill>
    </pool>
    <security>
        <user-name>user</user-name>
        <password>pass</password>
    </security>
</datasource>
```

Después de el datasource aparece los drivers al cual tenemos que añadir el de MySQL. Para este añadimos, después del ya existente, la siguiente línea:

```
<driver name="mysqlDriver" module="com.mysql">
<xa-datasource-class>com.mysql.jdbc.Driver</xa-datasource-
    class>
</driver>
```

Por último, vamos al apartado de security-domains donde están descritos los security-domain y añadimos una security-domain para nuestra aplicación:

```
<security-domain name="users">
<authentication>
    <login-module code="Database" flag="required">
        <module-option name="dsJndiName" value="java:/mysql
            "/>
```

```
        <module-option name="principalsQuery" value="SELECT
            password FROM user WHERE username = ?"/>
        <module-option name="rolesQuery" value="SELECT rol,
            'Roles' FROM user WHERE username = ?"/>
    </login-module>
</authentication>
</security-domain>
```

Una vez configurado el standalone.xml tenemos que añadir el .jar del driver de MySQL para que el servidor lo pueda utilizar. En este caso tenemos que entrar $WILDFLYPATH/modules/system/layers/base y crear la carpeta /com/mysql/main. Dentro de esta carpeta tenemos que meter el .jar del driver de MySQL que lo podemos descargar desde aquí:

http://central.maven.org/maven2/mysql/mysql-connector-java/5.1.38/mysql-connector-java-5.1.38.jar

También tenemos que añadir un fichero module.xml con el siguiente contenido:

```
<module xmlns="urn:jboss:module:1.1" name="com.mysql">
    <resources>
    <resource-root path="mysql-connector-java-5.1.38-bin.jar
        "/>
    </resources>
    <dependencies>
        <module name="javax.api"/>
        <module name="javax.transaction.api"/>
        <module name="javax.servlet.api" optional="true"/>
    </dependencies>
</module>
```

Por último, tenemos que añadir algunos archivos de la aplicación que al no poder configurar el path cuando lo leen y éste es un path relativo se tienen que añadir dentro de la carpeta del servidor. Es el caso de los archivos de las reglas para el conversor de BPMN a texto y los drivers del solver del comparador. Estos archivos los podemos encontrar en el repositorio en la carpeta webfiles/bin que hay en el trunk y los tenemos que dejar en la carpeta $WILDFLYPATH/bin.

A.4. Carpeta de configuración de la aplicación

Para organizar los archivos, la aplicación accede a una carpeta que tiene que estar en $HOME del usuario y se tiene que llamar 'BPMN'.

Dentro de esta carpeta tenemos que crear otras dos. En primer lugar una que se llame 'config' en la cual hay las carpetas de rule y words del conversor junto al archivo de las credenciales y las properties de la interfaz. La segunda carpeta se tiene que llamar 'exec' y contiene una carpeta que se llama ejemplos con dos subcarpetas que se llaman model y text donde se añaden los ejemplos de cada tipo. La carpeta exec también tiene otras dos carpetas llamadas 'BPMN y 'Text' donde se guardan los archivos que suben los usuarios. La estructura de carpetas que se ha explicado junto a los ficheros están en el repositorio en la carpeta webfiles del trunk y solo se tiene que copiar la carpeta BPMN entera.

Una vez tenemos la estructura de carpetas, necesitamos descargar el programa Petrify(Preferiblemente la version 4.1 para asegurar que es compatible) y dentro de la carpeta $HOME/BPMN/config encontraremos el archivo interface con las properties de la aplicación. Dentro de este archivo tenemos que cambiar la properties 'pathPetrify' y poner el path del archivo del Petrify que hemos descargado.

A.5. Despliegue de la aplicación

Para desplegar la aplicación debemos entrar vía web a localhost:9990 y entrar con el usuario que hemos creado al principio. Una vez dentro accedemos al apartado de 'Deployments' y apretar a 'Add' donde nos saldrá una ventana que nos preguntará si queremos crear un nuevo deploy. Cuando le demos a sí nos dará la opción de añadir un nuevo archivo, aquí tenemos que añadir el archivo bpmninterface.war que podemos encontrar en el repositorio y esperar a que se suba. Una vez acabe, ya podemos acceder a partir de la ip:8080/bpmninterface.

Referencias

[1] Camunda. https://bpmn.io. [Acceso: 29-02-2016].

[2] The Apromore Initiative. http://apromore.org/. [Acceso: 29-02-2016].

[3] Signavio. http://www.signavio.com/. [Acceso: 29-02-2016].

[4] https://en.wikipedia.org/wiki/Matching_(graph_theory). [Acceso: 20-06-2016].

[5] Michael Becker and Ralf Laue. A comparative survey of business process similarity measures. In *Computers in Industry*, January 2012.

[6] Marlon Dumas Luciano García-Bañuelos Abel Armas-Cervantes, Paolo Baldan. Behavioral comparison of process models based on canonically reduced event structures. In *Business Process Management: 12th International Conference, BPM 2014,*, September 2014.

[7] https://en.wikipedia.org/wiki/Apache$_{S}ubversion$. [$Acceso$: $22 - 06 - 2016$].

[8] Emili Sapena, Lluís Padró, and Jordi Turmo. Relaxcor: A global relaxation labeling approach to coreference resolution. In *Proceedings of the ACL Workshop on Semantic Evaluations (SemEval-2010)*, Uppsala, Sweden, July 2010.

[9] Camunda. https://camunda.com/bpm/features/, 2013. [Acceso: 10-06-2016].

[10] https://en.wikipedia.org/wiki/JavaServer_Faces. [Acceso: 29-02-2016].

[11] Red Hat. http://www.wildfly.org/, 2013. [Acceso: 01-06-2016].

[12] Oracle. https://www.mysql.com/, 2016. [Acceso: 15-06-2016].

[13] Universitat Politècnica de Catalunya. http://www.cs.upc.edu/ jordicf/petrify/, 1999. [Acceso: 12-06-2016].

[14] http://lpsolve.sourceforge.net/5.5/. [Acceso: 15-06-2016].

[15] Princeton University. https://wordnet.princeton.edu/, 2015. [Acceso: 18-06-2016].

[16] Activiti. http://activiti.org/. [Acceso: 10-06-2016].